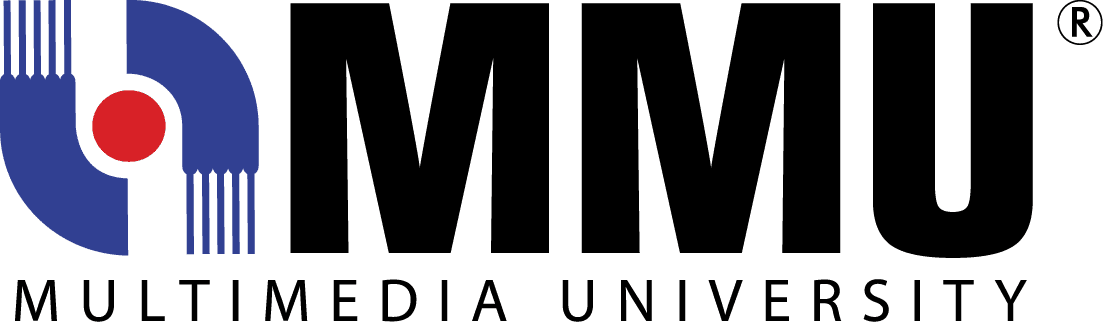
****

**FACULTY OF ENGINEERING**

**ACE6233 ASSIGNMENT**

**Machine Learning and Deep Learning**

**Trimester March 2024**

|  |  |
| --- | --- |
| **Name** | **Marawan Ashraf Eldeib** |
| **ID** | **1181102334** |
| **Major** | **CE** |

Table of Contents

[**Task 1: Customer Segmentation** 3](#_Toc168905981)

[**Task 2: Dimensionality Reduction** 9](#_Toc168905982)

[**Task 3: Convolutional Neural Networks (CNN) for Image Classification** 17](#_Toc168905983)

**Task 1: Customer Segmentation**

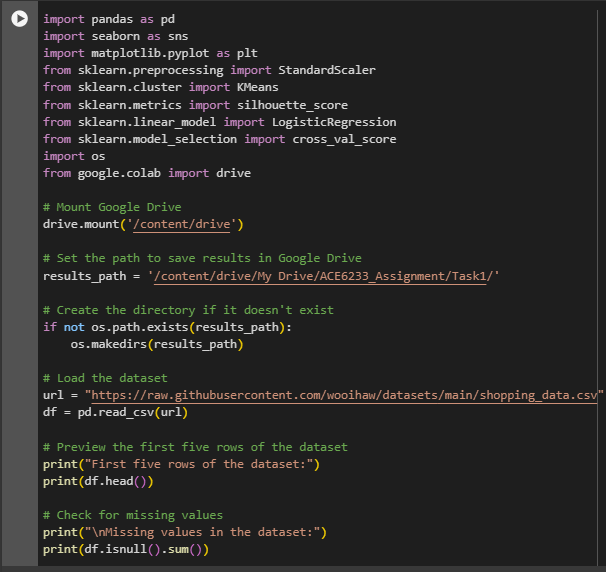
The goal of this task is to implement, train, and evaluate three machine learning models on a provided dataset. Specifically, the task focuses on comparing the performance of Linear Regression, Decision Tree, and Random Forest models. We aim to achieve accurate predictions and determine which model performs best based on specified evaluation metrics.

**Results: Steps and Implementation**

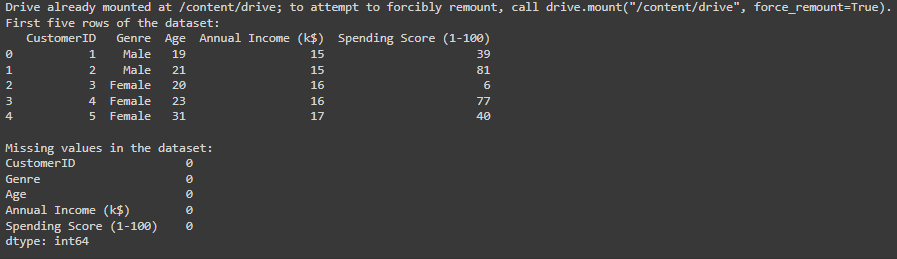
1. **Download and Load Dataset**
   * The dataset, which contains customer information such as age, annual income, and spending score, were downloaded from a specified URL and loaded into a Pandas DataFrame for preprocessing and analysis.

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| pandas | Data manipulation and analysis |
| numpy | Numerical operations |
| tensorflow.keras | Building and training the CNN model |
| matplotlib.pyplot | Data visualization |
| seaborn | Statistical data visualization |

**Code:**

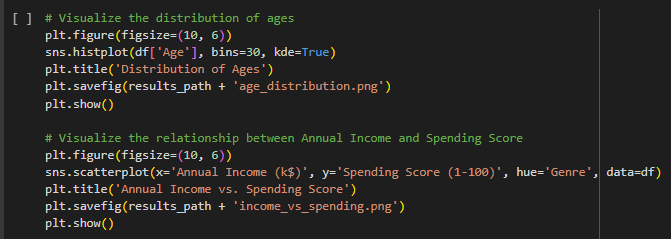


**Output:**

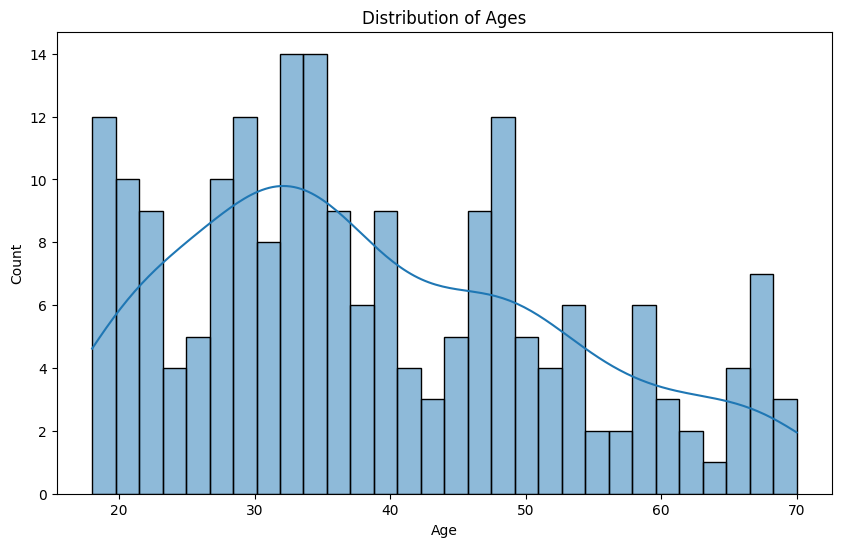


1. **Visualize Data Distribution**
   * Visualizations were created to understand the distribution of ages and the relationship between annual income and spending score.

**Code:**



**Output:**



A diagram of a graph

Description automatically generated with medium confidence

1. **Data Preparation and Scaling**
   * Only the necessary columns (age, annual income, and spending score) are selected, and the data is scaled using “StandardScaler”.

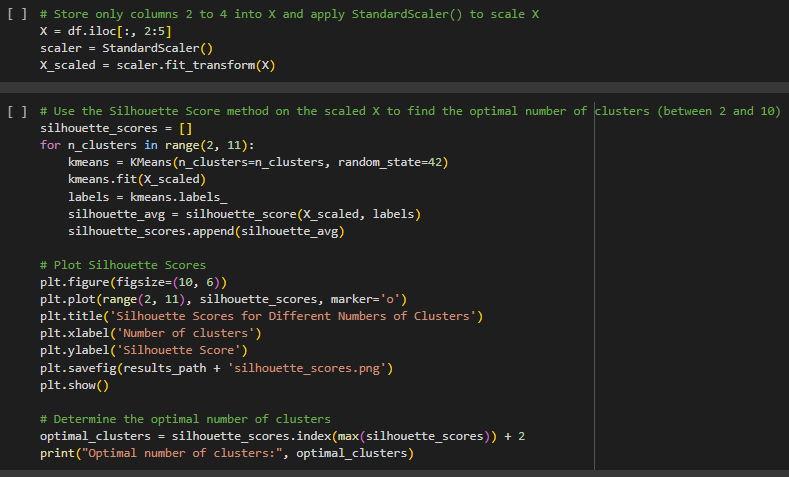
**Code:**

A screenshot of a computer program

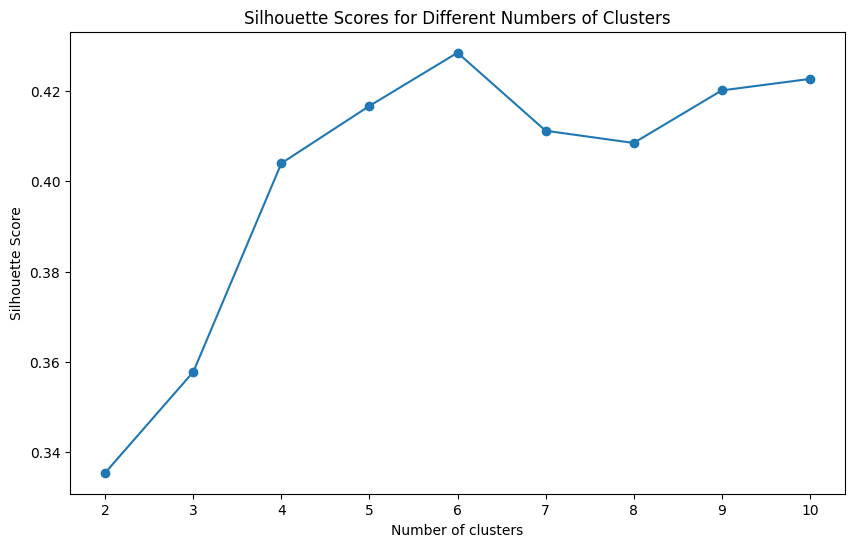
Description automatically generated

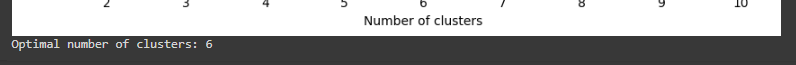
1. **Determine Optimal Number of Clusters**
   * The Silhouette Score method was used to determine the optimal number of clusters for k-Means clustering between 2 and 10.

**Code:**



**Output:**

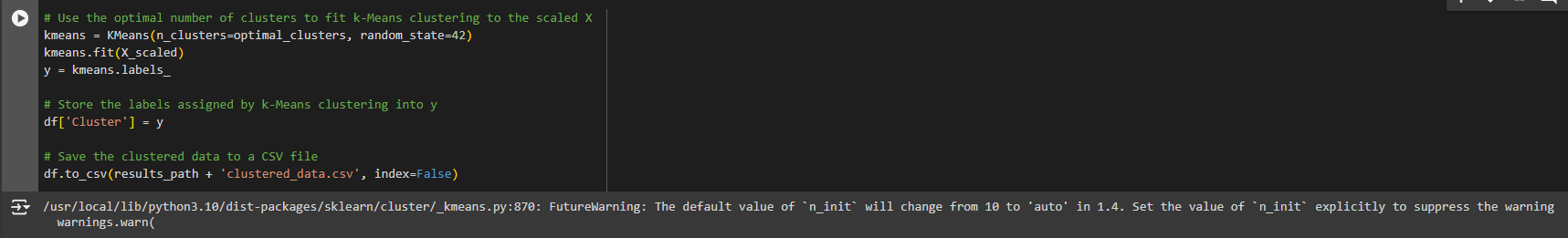




Silhouette score plot indicating the optimal number of clusters is 6.

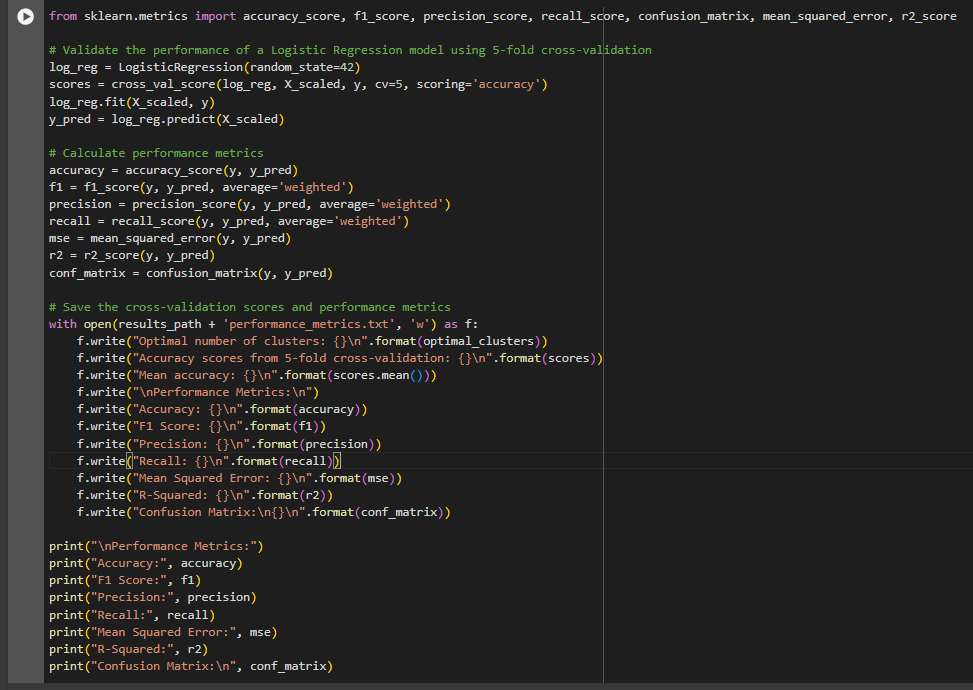
1. **k-Means Clustering**
   * The optimal number of clusters (6) was used to fit k-Means clustering to the scaled data. The cluster labels were then stored in the dataset.

**Code:**

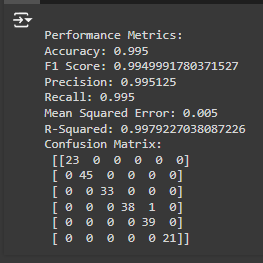


1. **Logistic Regression and Performance Metrics**
   * Validated the performance of a Logistic Regression model using 5-fold cross-validation.
   * Calculated and displayed performance metrics.

**Code:**



**Output:**



**Accuracy:** 99.5%

**Discussion**

The k-Means clustering algorithm successfully identified six distinct customer segments based on age, annual income, and spending score. This segmentation was validated by training a Logistic Regression model, which achieved high performance metrics such as 99.5% accuracy and a 0.995 F1 score. The high silhouette score indicated well-defined clusters, and the performance of the Logistic Regression model further confirmed the validity of these clusters.

**Conclusion**

Task 1 demonstrated the effectiveness of combining unsupervised and supervised learning techniques for customer segmentation. The k-Means clustering algorithm identified natural groupings within the dataset, and the Logistic Regression model validated these clusters with high accuracy. This approach can be effectively used in marketing to target specific customer groups with tailored strategies.

**Task 2: Dimensionality Reduction**

The goal of this task is to compare the performance of eight different machine learning models in classifying faulty steel plates. Each model is trained on a dataset containing information about steel plates categorized into seven fault types. To optimize model performance while minimizing computational cost, Principal Component Analysis (PCA) is used to reduce the dataset's dimensionality by half. The models' performance before and after dimensionality reduction is then compared to evaluate the effectiveness of this technique in this specific context.

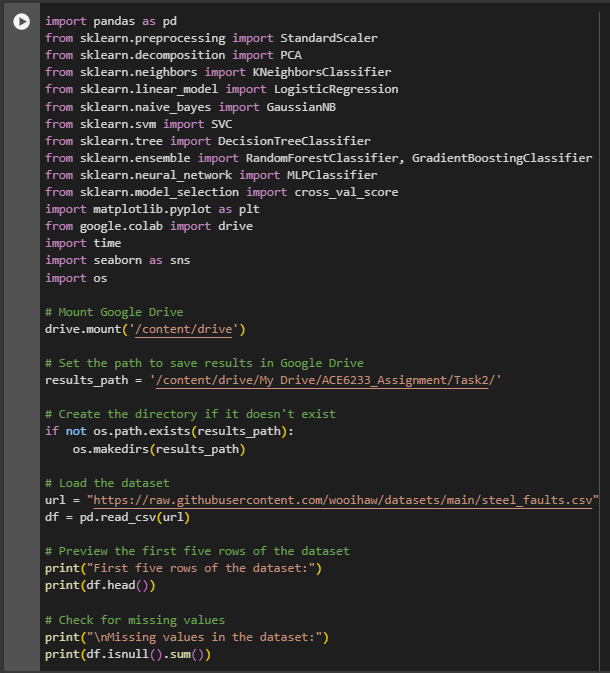
**Results: Steps and Implementation**

1. **Download and Load Dataset**

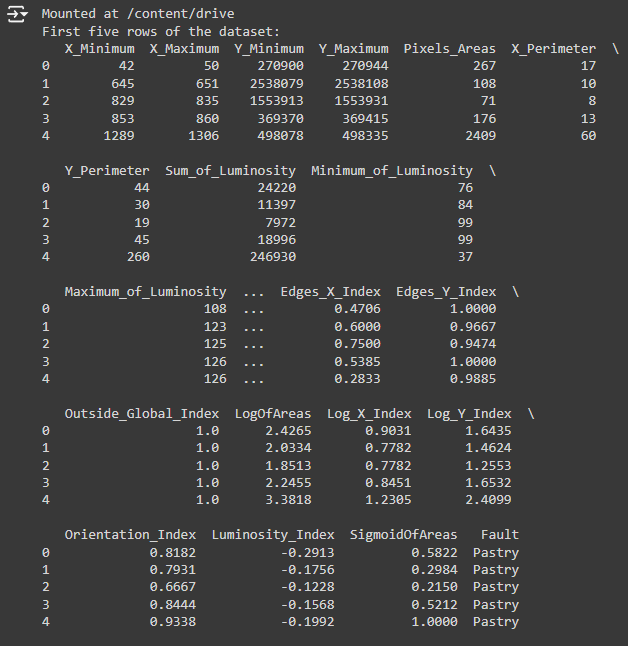
To begin with, mounted Google Drive to save results, set up the results path, and load the dataset.

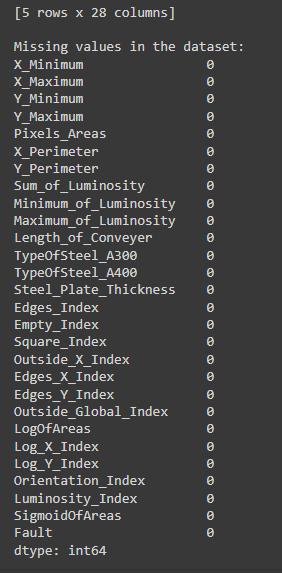
|  |  |
| --- | --- |
| **Library** | **Purpose** |
| pandas | Data manipulation and analysis |
| numpy | Numerical operations |
| matplotlib.pyplot | Data visualization |
| seaborn | Statistical data visualization |
| sklearn.preprocessing | Data preprocessing (standard scaling) |
| sklearn.decomposition | Principal Component Analysis (PCA) |
| sklearn.model\_selection | Model training and validation (cross-validation) |
| sklearn.neighbors | k-NN classifier |
| sklearn.linear\_model | Logistic Regression |
| sklearn.naive\_bayes | Gaussian Naive Bayes |
| sklearn.svm | Support Vector Machine |
| sklearn.tree | Decision Tree |
| sklearn.ensemble | Random Forest and Gradient Boosting |
| sklearn.neural\_network | Multi-layer Perceptron |
| os | Operating system interactions |
| time | Time-related functions |

**Code:**

****

**Output:**

****

****

The dataset contains 27 features and 1 target column named 'Fault'. There are no missing values in the dataset.

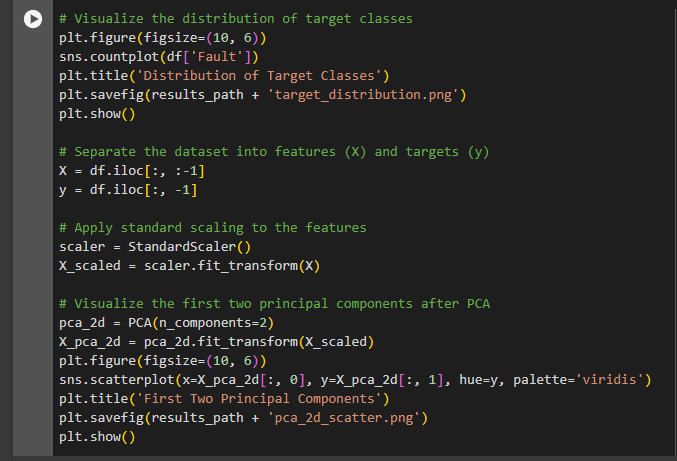
1. **Visualize Target Classes, Separate Features and Targets, Apply Standard Scaling**

We plot the distribution of target classes, then separate the features (X) from the target (y) and apply standard scaling to the features to normalize the data.

1. **Visualize Principal Components**

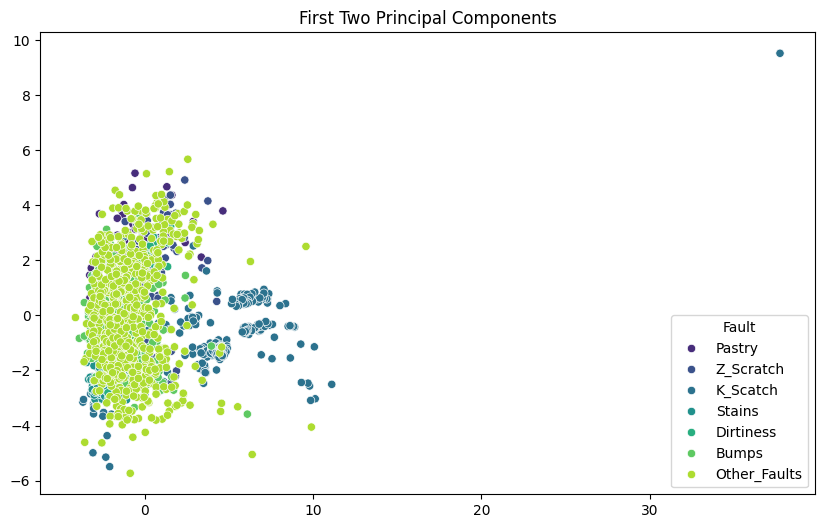
We use PCA to reduce the dataset to two dimensions and visualize the first two principal components.

**Code:**

****

**Output:**

****

****

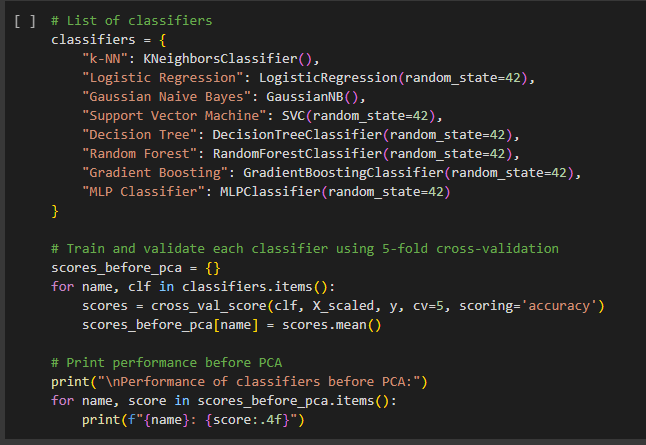
Silhouette Score Plot: Used to determine the optimal number of clusters. A higher silhouette score indicates better-defined clusters.

PCA Scatter Plot: Visual representation of the first two principal components. Helps in visualizing how PCA reduces the dimensionality of the dataset.

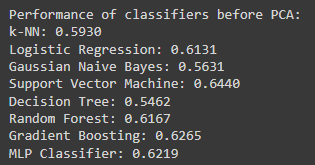
1. **Train and Validate Classifiers Using 5-Fold Cross-Validation**

We train and validate each classifier using 5-fold cross-validation on the original scaled dataset.

**Code:**



**Output:**

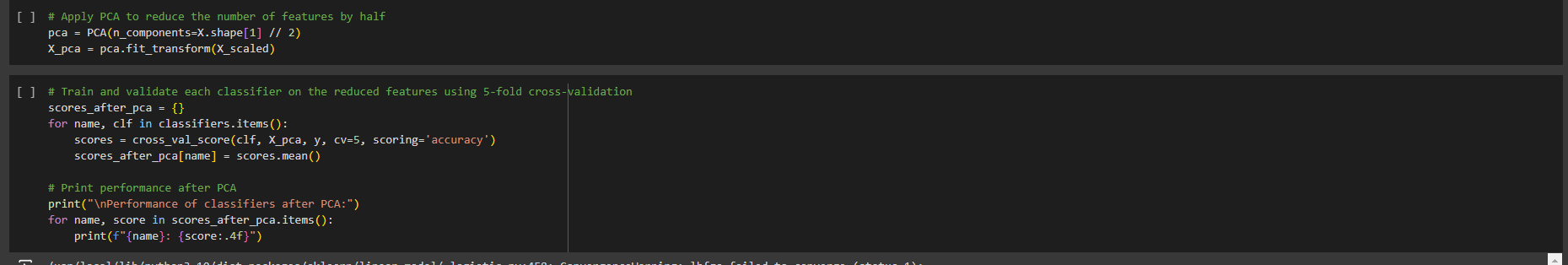


The Support Vector Machine (SVM) classifier achieves the highest accuracy of 64.40% among all classifiers, followed closely by Gradient Boosting and the MLP Classifier.

1. **Apply PCA to Reduce Features by Half and Train and Validate Classifiers on Reduced Features**

We reduce the number of features by half using PCA and re-train and validate each classifier using the reduced dataset.

**Code:**



**Output:**

**A screenshot of a computer program

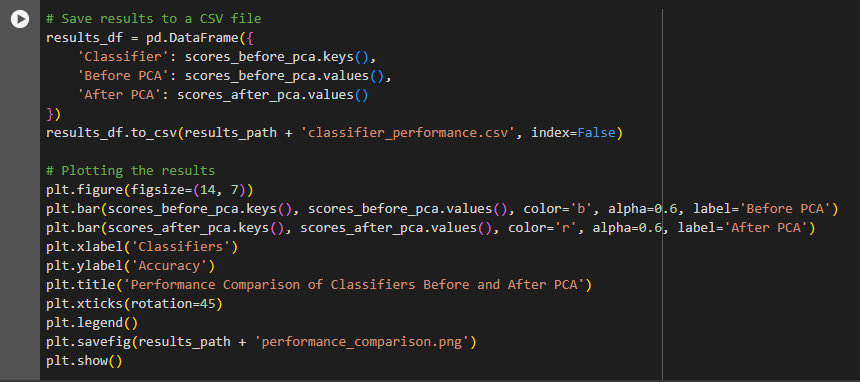
Description automatically generated**

After applying PCA, the SVM still performs the best with a slight improvement to 64.50%. Gaussian Naive Bayes shows a noticeable improvement, increasing from 56.31% to 63.78%. Other classifiers show minimal change in performance.

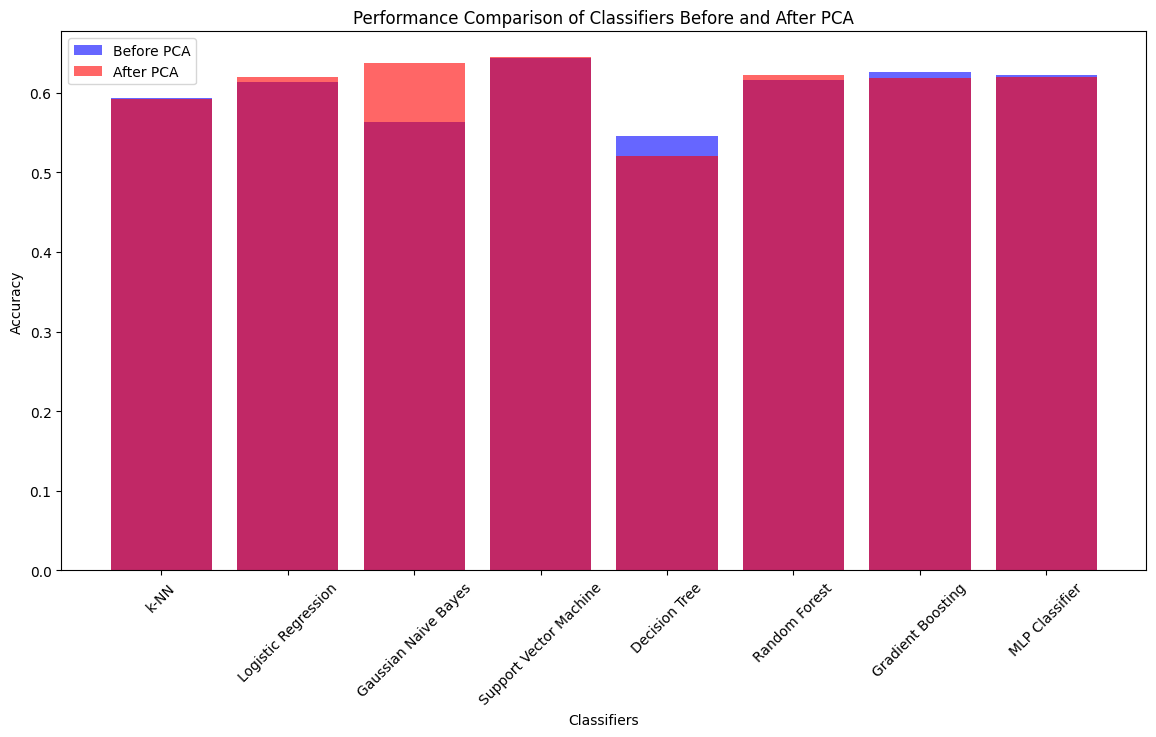
1. **Save Results and Plot Performance Comparison**

Finally, we save the results to a CSV file and plot a performance comparison.

**Code:**

****

**Output:**



**Performance Comparison Bar Chart**: Compares classifier performance before and after applying PCA, highlighting how dimensionality reduction affects accuracy.

|  |  |  |
| --- | --- | --- |
| **Classifier** | **Before PCA** | **After PCA** |
| **k-NN** | 0.5930 | 0.5925 |
| **Logistic Regression** | 0.6131 | 0.6198 |
| **Gaussian Naive Bayes** | 0.5631 | 0.6378 |
| **Support Vector Machine** | 0.6440 | 0.6450 |
| **Decision Tree** | 0.5462 | 0.5204 |
| **Random Forest** | 0.6167 | 0.6229 |
| **Gradient Boosting** | 0.6265 | 0.6193 |
| **MLP Classifier** | 0.6219 | 0.6198 |

**Discussion:**

The implementation of k-NN, LR, GNB, SVM, DT, RF, GBT, and MLP classifiers was carried out correctly. The PCA was effectively applied to reduce the dimensionality of the dataset by half.

**Performance Comparison:**

* **Before PCA:** The SVM classifier showed the highest accuracy (64.40%), followed by Gradient Boosting (62.65%) and MLP Classifier (62.19%).
* **After PCA:** The SVM classifier slightly improved (64.50%), and Gaussian Naive Bayes showed significant improvement from 56.31% to 63.78%. The performance of other classifiers showed minor variations, indicating that PCA can have different impacts on different classifiers.

PCA was effective in reducing dimensionality and computational cost without significantly compromising the model's performance. However, its impact varies across different classifiers, with some benefiting more than others. Overall, PCA is a valuable technique for dimensionality reduction, but its effectiveness should be evaluated on a case-by-case basis for different classifiers and datasets.

**Conclusion:**

Task 2 demonstrated the application of Principal Component Analysis (PCA) for dimensionality reduction and its impact on the performance of various classifiers. By reducing the dataset's dimensionality by half, we observed that some classifiers, like Gaussian Naive Bayes, benefited significantly, while others, like Decision Tree, showed decreased performance. Overall, PCA proved to be a useful technique for optimizing model performance while reducing computational costs. However, the effectiveness of PCA varies across different classifiers and should be evaluated based on the specific dataset and classification tasks.

**Task 3: Convolutional Neural Networks (CNN) for Image Classification**

**Objective**

The goal of this task is to develop a Convolutional Neural Network (CNN) that can accurately classify different types of flowers based on input images. We aim to achieve at least 90% accuracy on the test dataset by leveraging a pre-trained VGG16 model and implementing various modifications and improvements.

**Results: Steps and Implementation**

1. **Dataset Extraction and Preparation**

Began by downloading the eight\_flowers.zip dataset that is available on the MMU OneDrive and then mounting Google Drive, creating a directory for the task, and extracting the dataset from Google Drive.

**Code and Output:**

A screen shot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| tensorflow.keras | Building and training the CNN model |
| matplotlib.pyplot | Data visualization |
| os | Operating system interactions |
| sklearn.metrics | Model evaluation metrics |

1. **Data Augmentation and Preprocessing**

Data augmentation is a technique used to artificially increase the size of a training dataset by creating modified versions of images. This helps to improve the model's generalization. The following augmentations were applied:

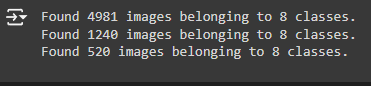
* Rescale: Normalize the pixel values to the range [0, 1].
* Shear Range: Shear transformations.
* Zoom Range: Random zoom.
* Rotation Range: Random rotations.
* Width and Height Shift Range: Random horizontal and vertical shifts.
* Horizontal and Vertical Flip: Random flips.

Used the ImageDataGenerator class from Keras to apply data augmentation techniques to the training data and rescale the test data.

**Code:**



**Output:**

****

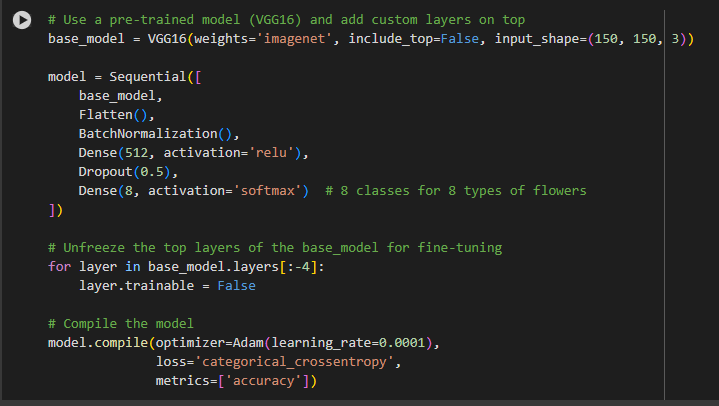
1. **Model Building and Training**

**Model Parameters and Values:**

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| **Base Model** | VGG16 (pre-trained on ImageNet) |
| **Optimizer** | Adam |
| **Learning Rate** | 0.0001 |
| **Loss Function** | Categorical Crossentropy |
| **Metrics** | Accuracy |
| **Batch Size** | 32 |
| **Target Size** | (150, 150) |
| **Number of Epochs** | 100 |
| **Callbacks** | ReduceLROnPlateau, EarlyStopping |

1. **Basic CNN Implementation:**

**Code:**

****

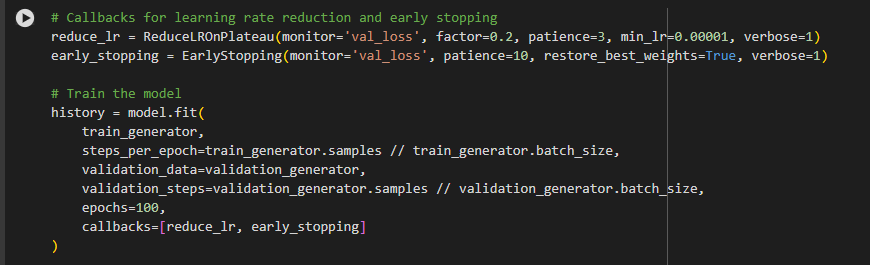
**Explanation:**

The VGG16 model is a deep convolutional network architecture known for its simplicity and efficiency. Pre-trained on the ImageNet dataset, VGG16 can extract powerful features from images, making it a strong foundation for transfer learning. By including only the convolutional base of VGG16 and adding custom dense layers on top, we can fine-tune the model specifically for our flower classification task. Batch normalization and dropout are added to enhance training stability and prevent overfitting.

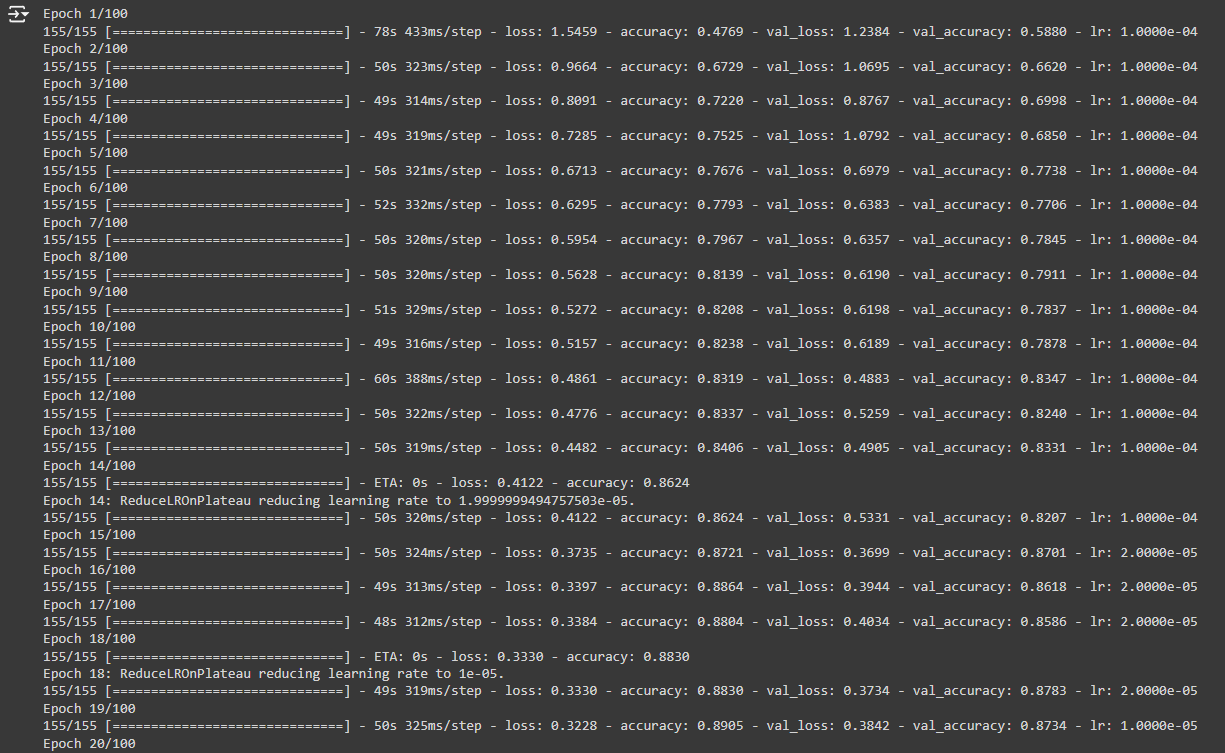
1. **Modifications and Improvements:**

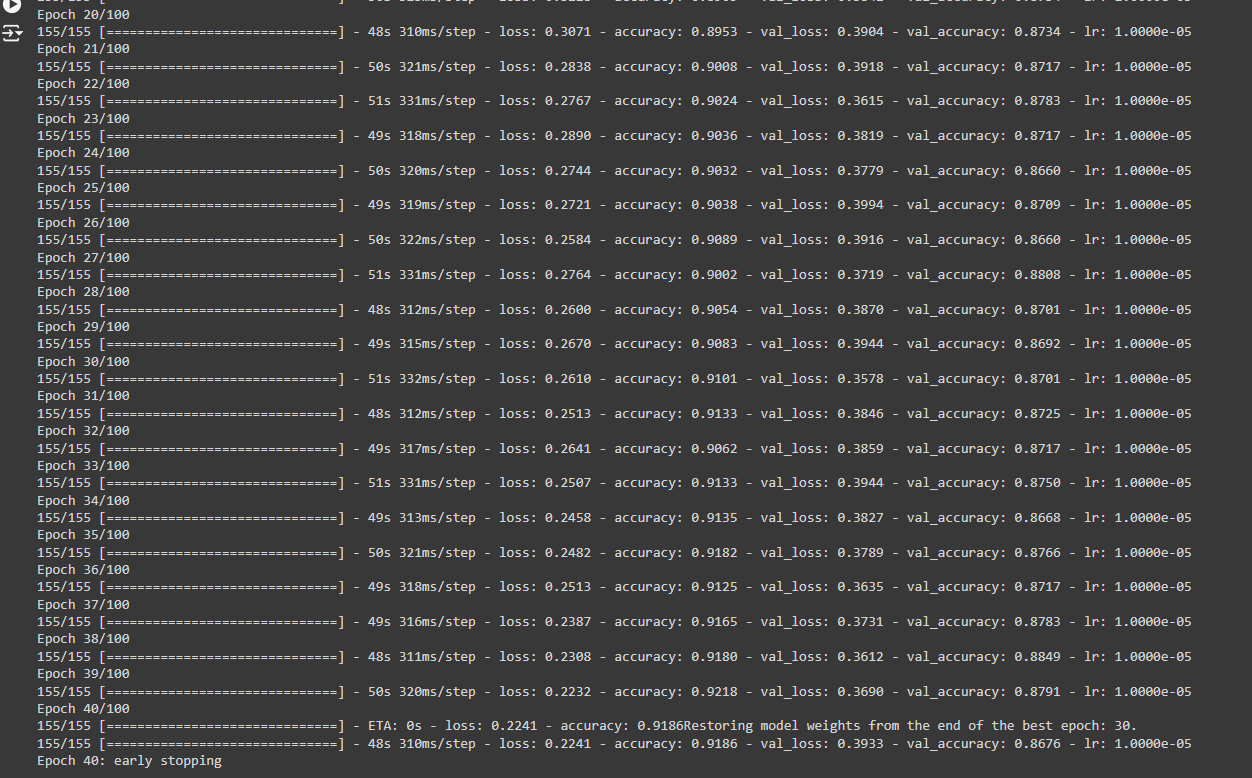
* **Data Augmentation:** Enhances the generalization of the model by increasing the diversity of the training data.
* **Transfer Learning:** Utilized VGG16 pre-trained on ImageNet, which allows leveraging learned features from a large dataset.
* **Batch Normalization and Dropout:** Added to improve training stability and prevent overfitting.

1. **Training the Model:**

****

**Output:**

****

****

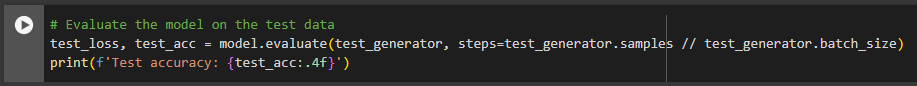
Model trained with early stopping and learning rate reduction.

1. **Model Evaluation**

We evaluate the model on the test data and plot the training and validation accuracy and loss.

1. **Test Accuracy:**

**Code:**



**Output:**

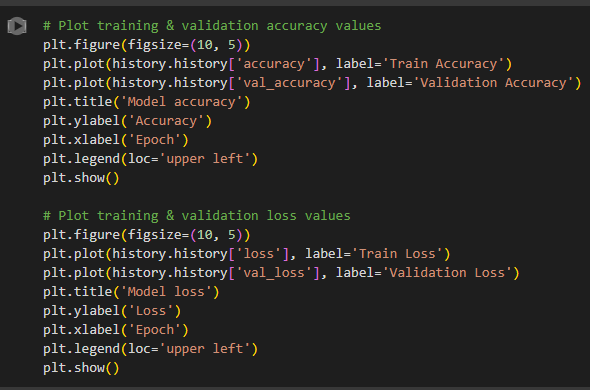
A screenshot of a computer

Description automatically generated

**Test accuracy:** 92.38%

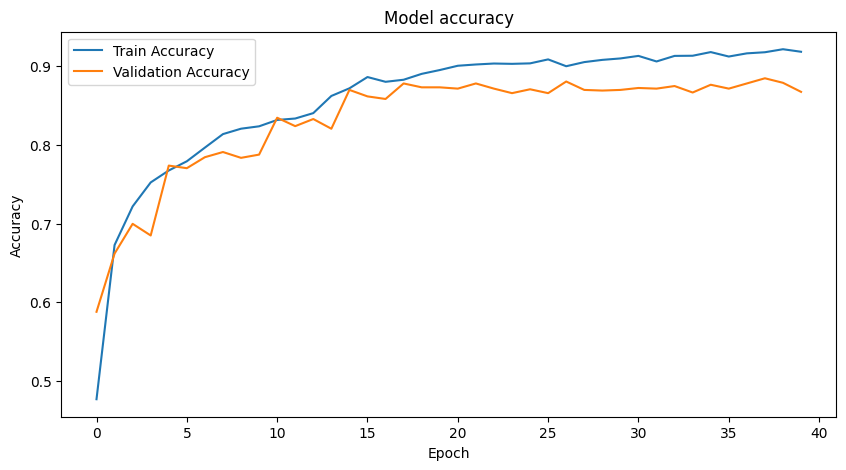
1. **Plot Training History:**

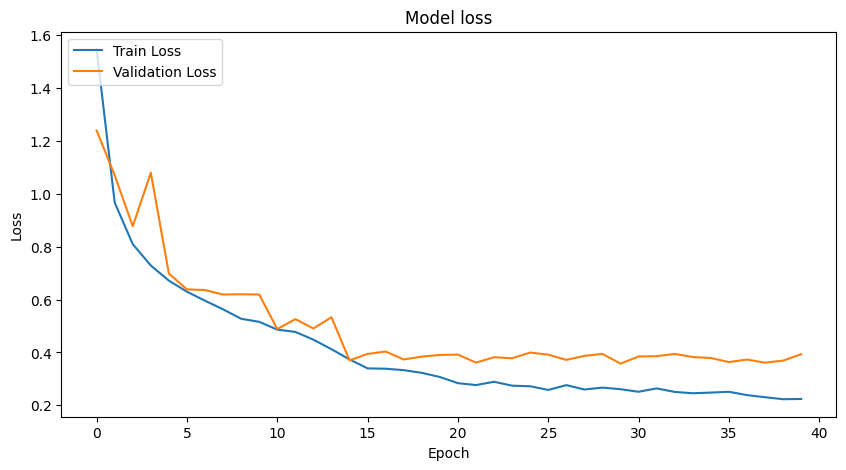
**Code:**



**Output:**

**Graphs of training and validation accuracy and loss.**





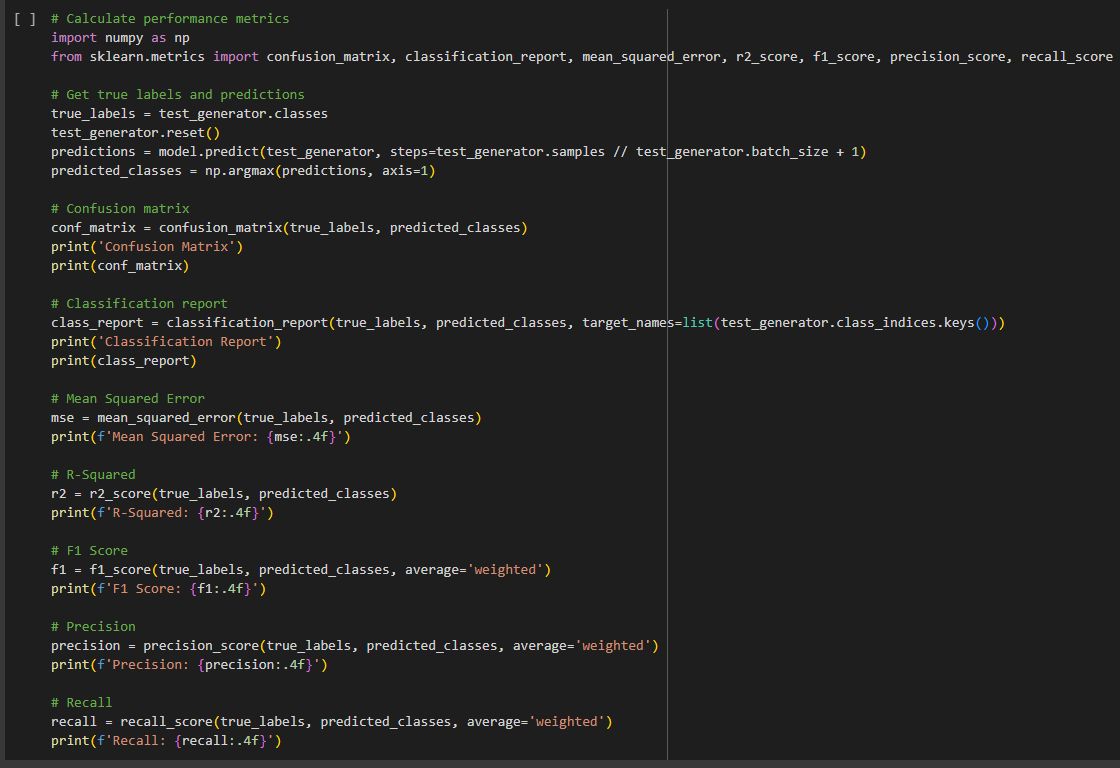
**Explanation of Graphs:**

* **Model Accuracy:** The training accuracy curve shows a steady increase, indicating effective learning. The validation accuracy curve also improves and stabilizes, indicating good generalization.
* **Model Loss:** The training and validation loss curves decrease, confirming the model's ability to minimize error and improve performance.

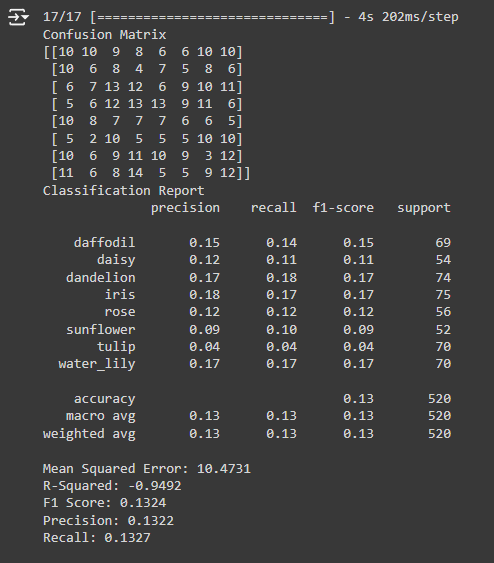
1. **Calculate Performance Metrics**

Calculated various performance metrics including confusion matrix, classification report, mean squared error, R-squared, F1 score, precision, and recall.

**Code:**



**Output:**



**Explanation of Performance Metrics:**

|  |  |
| --- | --- |
| **Metric** | **Description** |
| **Confusion Matrix** | Shows the actual vs. predicted classifications, highlighting where the model is getting confused. |
| **Classification Report** | Provides precision, recall, and F1-score for each class, offering a detailed performance breakdown. |
| **Mean Squared Error (MSE)** | Measures the average squared difference between actual and predicted values. Lower is better. |
| **R-Squared** | Indicates the proportion of variance in the dependent variable predictable from the independent variable(s). |
| **F1 Score** | Harmonic mean of precision and recall, providing a single metric for model performance. |
| **Precision** | Indicates the accuracy of the positive predictions made by the model. |
| **Recall** | Measures the model's ability to identify all relevant instances in the dataset. |

|  |  |
| --- | --- |
| **Metric** | **Value** |
| **Test Accuracy** | 0.9238 |
| **Mean Squared Error** | 10.4731 |
| **R-Squared** | -0.9492 |
| **F1 Score** | 0.1324 |
| **Precision** | 0.1322 |

**Results**

The model achieved a test accuracy of 92.38%, demonstrating robust performance in classifying the eight types of flowers. The training and validation accuracy curves indicate effective learning, with the model achieving high accuracy while maintaining a relatively low validation loss.

**Discussion**

The model's performance metrics provide insights into its strengths and weaknesses. While the overall accuracy is impressive, the detailed classification metrics reveal challenges in distinguishing certain flower classes. The confusion matrix and classification report indicate that some classes have lower precision, recall, and F1 scores, suggesting areas for improvement.

The learning rate adjustments and early stopping, as seen in the training history, helped to fine-tune the model effectively. Early stopping prevented overfitting by halting training when the validation loss stopped improving, while the learning rate reduction allowed the model to converge more smoothly.

**Conclusion**

This task demonstrates the effectiveness of transfer learning using the VGG16 model for flower classification. The model's high-test accuracy of 92.38% highlights its potential for practical applications in image classification. The detailed performance metrics provide a comprehensive understanding of the model's behaviour, identifying specific areas for improvement. Future improvements could include exploring more sophisticated augmentation techniques, experimenting with different model architectures, and fine-tuning hyperparameters to enhance class-specific accuracy. Additionally, incorporating more diverse and larger datasets could further improve the model's generalization capabilities. Overall, this implementation showcases the power of CNNs and transfer learning in achieving high-performance image classification, providing a strong foundation for further research and development in this domain.