TRANSFER LEARNING USING PISTACHIO IMAGE DATASET

Semester – 7

Major Project – I (01CE0716)

A PROJECT REPORT

Submitted by

HELI HATHI

92100103341

HET BUCH

92100103196

BACHELOR OF TECHNOLOGY

in

Computer Engineering

Marwadi University, Rajkot

December, 2024

Major Project-I (01CE0716)

Marwadi University

Faculty of Technology

Department of Computer Engineering

2024-25

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **TRANSFER LEARNING USING PISTACHIO IMAGE DATASET** has been carried out by **Heli Hathi** (92100103341) and **Het Buch** (92100103196) under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 7th Semester of Marwadi University, Rajkot during the academic year 2024-25.

Prof. Ravikumar R Natarajan	Dr. Krunal Vaghela
Assistant Professor	Associate Professor
Internal Guide	Head of the Department

Marwadi University
Marwadi Chandarana Group
NAAC A+

Marwadi University
Rajkot

DECLARATION

We hereby declare that the **Major Project-I (01CE0716)** report submitted along with the Project entitled **Transfer Learning using Pistachio Image Dataset** submitted in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering to Marwadi University, Rajkot, is a bonafide record of original project work carried out by us at Marwadi University under the supervision of **Prof. Ravikumar R Natarajan** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

	Name of the Student	Sign of Student
1	______________________	_______________
2	______________________	_______________

Acknowledgement

We want to express our gratitude to our mentor, Professor Ravikumar Natrajan, for his important advices and assistance during the project. The effective completion of the project was greatly influenced by his oversight and insightful observations.

We also owe a gratitude to Dr. Krunal Vaghela, the head of Marwadi University Department of Computer Engineering, for his constant support and for encouraging a collaborative atmosphere.

We sincerely thank our friends for their constant support, feedbacks, and insightful comments.

Heli Hathi,

Het Buch.

Abstract

Pistachio nuts, a valuable nutritional source widely cultivated for commercial purposes, come in two distinct varieties: Kirmizi and Siirt. Accurate classification of these varieties is crucial for maintaining product quality, value, and facilitating trade, as they differ in price, taste, and nutritional content. However, manual classification presents challenges such as inconsistency due to subjective interpretation, and labour-intensive processes requiring significant human resources. To address these issues, various machine learning models have been applied to identify pistachio types, particularly for unpackaged nuts. Transfer learning (TL) is a promising approach to reduce the time and effort required for developing learning models across different classification problems. This project uses common TL architectures, including ResNet, VGG, DenseNet, AlexNet, and SqueezeNet, to create a pistachio classification application using FastAI library and also Xception, EfficientNet and RegNetX using Timm library. Through experimentation with both FastAI and Timm models, we achieved highly accurate results, with ResNet34, VGG19_bn and Efficientnet_b2 achieving accuracies of 99.5. These models aim to overcome the challenges associated with implementing a classifier for unpackaged pistachio products, enhancing the efficiency and accuracy of the sorting process in the food industry. Furthermore, we expand this project to implement in real-time by creating a web-based application using Streamlit, a Python framework and integrate the best performing model for a better user experience and pistachio classification.

List of Figures

List of Tables

Abbreviations

TL	**Transfer Learning**
bn	**Batch Normalized**
ResNet	**Residual Network**
VGG	**Visual Geometry Group**
DenseNet	**Densely Connected Convolutional Network**
Xception	**Extreme Inception**
RegNetX	**Regularized Network X**
Timm	**PyTorch Image Models**
AdaBoost	**Adaptive Boosting**
CNN	**Convolutional Neural Network**
KNN	**K Nearest Neighbour**
YOLO	**You Only Look Once**
ML	**Machine Learning**
CPU	**Central Processing Unit**
GPU	**Graphics Processing Unit**
Eqdiasq	**Equivalent Diameter Square**
Std_Dev	**Standard Deviation**
RGB	**Red, Green, Blue**
CatBoost	**Categorical Boosting**
AUC	**Area under the Curve**
ROC	**Receiver Operating Characteristics**

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Project Summary

Pistachios are widely farmed commercially in two varieties: Kirmizi and Siirt. However, manual classification has drawbacks such as inconsistency due to individual interpretation and labour-intensive methods that need large human resources. To overcome these challenges, multiple machine learning models have been used to detect different varieties of pistachios, particularly unpacked nuts. Transfer learning is a potential strategy to reducing the time and effort required for developing models for a variety of classification tasks. TL models like ResNet, VGG, DenseNet, AlexNet, and SqueezeNet from FastAI, as well as Xception, EfficientNet, and RegNetX from the Timm library. These models produced promising and accurate results using both FastAI and Timm models, out of which ResNet34, VGG19_bn, and Efficientnet_b2 reached the accuracies of 99.5%. Furthermore, we extended this project to real-time by developing a web-based application with Streamlit, a Python framework, and incorporating the best-performing model for a real time pistachio classification.

1.2 Purpose

Pistachios, particularly the Kirmizi and Siirt varieties, hold significant economic value in agriculture. The increasing demand for accurate classification across industries necessitates efficient automatic sorting methods [1][4]. Traditional approaches are becoming inadequate, prompting the exploration of advanced technologies. Recent advancements in deep learning, especially transfer learning, offer promising solutions leveraging pre-trained neural network models to reduce resource requirements while maintaining high accuracy [9].

1.3 Objective

This study aims to develop an innovative classification model using image processing and artificial intelligence, other than conventional separation methods. The goal is to improve efficiency in pistachio sorting, supporting informed decision-making and sustainable resource management in the agricultural sector [5].

1.4 Scope

The scope of this project is to create an automatic classification system for two different types of pistachios, Kirmizi and Siirt, using transfer learning techniques. The project utilizes both numerical feature analysis (with 16 and 28 feature datasets) and image-based classification using multiple pre-trained deep learning models such as ResNet, VGG, DenseNet, and EfficientNet models. This technology has major uses in the agriculture business, particularly in and sorting techniques, where it can replace manual classification techniques with a more efficient, consistent, and scalable automatic approach. Despite these advantages, it requires careful selection of base models and fine-tuning strategies. Furthermore, the system may find challenging differentiating between varieties when pistachios has too much similarity, potentially affecting classification accuracy in real-world industrial settings.

1.5 Literature Review

Recent research in pistachio classification demonstrates diverse approaches and promising results across different methodologies. A thorough review of recent research, is shown in Table 1.1. Ref. [1] implemented transfer learning using ResNet, EfficientNet, Inception, and MobileNet, achieving 96% accuracy while reducing model training effort, though faced challenges with real-time validation. Ref. [2] utilized VGG16 and achieved remarkable 99.1% accuracy by leveraging CNNs for complex pattern recognition, despite working with a limited dataset size. Ref. [3] employed VGG16 and VGG19 deep learning architectures, achieving 80.21% accuracy, though noted that image pre-processing could improve accuracy. Ref. [7] focused on food safety detection using pre-trained CNN models, demonstrating strong robustness and adaptability to practical scenarios with 92% accuracy, though challenges remained in food safety aspects despite good classification performance. Ref. [4] employed KNN classification, reaching 94.18% accuracy with reduced time and energy expenses, though lacking a training process. Notably, Ref. [5] developed a hybrid approach combining ResNet21 and DenseNet, achieving 99.3% accuracy with enhanced precision and loss reduction to 0.0137, although dataset restrictions limited its generalizability. More recent work by Ref. [6] using EfficientNetB3 achieved 99.5% accuracy in pistachio classification, while Ref. [8] demonstrated 99% accuracy using a combination of AlexNet, VggNet, ResNet, and CNN models for pistachio recognition.

A notable mobile application implementation by Ref. [9] using InceptionV3 and ResNet50 achieved 96% accuracy, though faced challenges with similar-shaped pistachios. Ref. [10] explored YOLOv8 for detection tasks, achieving 94.8% mAP specifically for open and closed shell pistachios. Ref. [11] utilized ResNet achieving 88.57% accuracy with high commercial potential, though lacking comparative analysis with other methods.

Table 1.1 Literature Review of Previous Works

Ref No.	**Dataset**	**Algorithms**	**Accuracy**	**Merits**	**Demerits**
[1]	Custom Dataset	ResNet, EfficientNet, Inception, and MobileNet	**96%**	TL reduces effort for model training on fewer instances.	Difficulty in classifying due to size and lack of real-time validation.
[2]	Kaggle	VGG16	**99.1%**	CNNs learn visuals, suitable for finding complex patterns.	Small dataset size for training CNN models.
[3]	Custom Dataset	VGG16, VGG19	**80.21%**	Effective classification with DL architectures.	Image pre-processing could improve accuracy.
[4]	Custom Dataset	KNN	**94.18%**	Reduced time and energy expenses.	Lacks training process.
[5]	Kaggle	Hybrid ResNet21, DenseNet	**99.3%**	Enhanced precision and reduced the loss to 0.0137.	Dataset restrictions limit accessibility and generalizability.
[6]	Kaggle	EfficientNet B3	**99.4%**	Enhanced pistachio classification.	Limited dataset size affects model performance.
[7]	Kaggle	ResNet	**92%**	Strong robustness and adaptability to practical scenarios.	Challenges in food safety despite good performance.
[8]	Custom Dataset	AlexNet, VggNet, ResNet, CNN	**99%**	Enhanced accuracy and efficiency in pistachio classification.	Model refinement needed to enhance performance.
[9]	Kaggle	InceptionV3, ResNet50	**96%**	Mobile app for quick pistachio type determination in agriculture.	Identical form of pistachio increases classification mistakes.

[10]	Pesteh Dataset	YOLOv8	**mAP (94.8%)**	Efficient detection of open and closed shell pistachios.	Limited to YOLOv8 only, and no other detection models.
[11]	Custom Dataset	ResNet	**88.57%**	Accurate classification of pistachio with high commercial potential.	No comparison with other methods makes it hard to assess its efficiency.

1.6 Project Scheduling

Our project followed a thorough method to ensure effective execution and timely completion. We organized project stages from the start of the Data Collection to the completion of the web-application. This provided clear visualization and use of resources, resulting in effective teamwork. In addition, regular team meetings with our project guide allowed open communication, cooperation, and timely solution of any potential problems Fig. 1.1 shows the Gantt chart of our project scheduling.

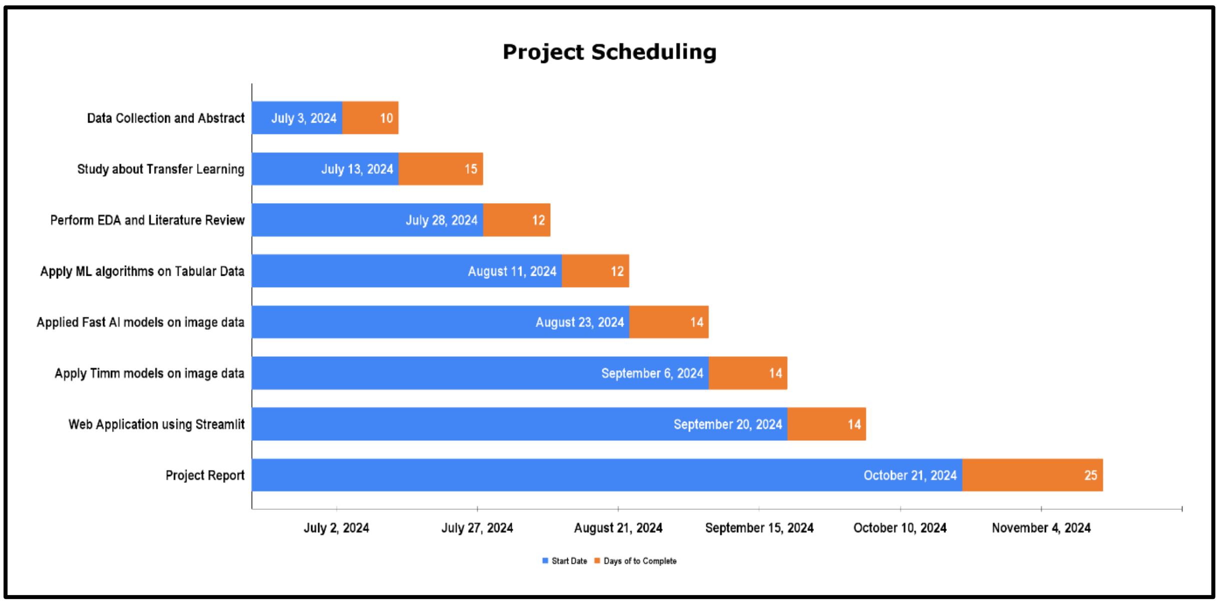


Fig. 1.1 Visualization of Project Planning using Gannt Chart

CHAPTER 2

SYSTEM ANALYSIS

2.1 Study of the current system

The current pistachio classification system primarily relies on traditional manual methods involving subjective visual inspection and evaluation by human experts. Researches highlight a range of techniques that have been explored to address the limitations of this manual approach. These include the application of machine learning algorithms such as KNN as well as computer vision-based methods leveraging pre-trained CNN architectures like VGG, ResNet, DenseNet, and Inception have also demonstrated promising results in accurately classifying pistachio varieties. Additionally, hybrid approaches combining multiple models have also been applied to enhance classification performance.

2.2 Problem and Weakness of Current System

Manual classification of different pistachio categories remains a major difficulty due to the wide variety and almost similar shape of pistachio nuts. Humans are susceptible to subjective biases, inconsistencies, and mistakes caused by factors such as personal judgment or tiredness. As a result of the high labour costs and limited output, large-scale pistachio manufacturers struggle to fulfil market demand. Furthermore, manual sorting can cause quality and pricing concerns since human mistakes in classification might reduce the overall quality and sale price of the product. While digital image processing has transformed many sectors, effectively categorizing crop results based on appearance is still a major challenge in this field. Also, limited dataset size, real-time validation can may all result in inaccurate classification. Given the economic significance of pistachios, there is a clear need for novel automated methods and smart technologies to enable efficient, accurate, and cost-effective pistachio classification and quality determination, eventually boosting consumption, marketing, and sustainable resource management.

2.3 Requirements of New System

The main objective of this project's new pistachio classification system is to use transfer learning techniques to differentiate between the Kirmizi and Siirt pistachio types with high accuracy. Using pre-trained deep learning models such as ResNet, VGG, DenseNet, and EfficientNet, the system tries to achieve high classification accuracy, outperforming previous attempts. The system is able to handle both numerical feature data (16 and 28 features) and image data from pistachio samples while effectively integrating computer vision and machine learning capabilities. Furthermore, the new system is built as a web application using the Streamlit framework, allowing for real-time classification and use in industries. This user-friendly interface provides clear results to help with quality control and sorting processes. By addressing these, the project aims to provide a powerful, automated solution that addresses the drawbacks of human pistachio classification and also meeting the demands of large-scale pistachio processors.

2.4 System Feasibility

2.4.1 Does the system contribute to the overall objectives of the organization?

Ans. Yes

2.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints?

Ans. Yes

2.4.3 Can the system be integrated with other systems which are already in place?

Ans. Yes

2.5 Proposed System

Machine learning is a field of artificial intelligence that enables computers to learn from and make predictions or decisions based on data. For this project, we applied various machine learning algorithms like Logistic Regression, KNN, Decision Tree, etc. as well as ensemble algorithms like AdaBoost, Gradient Boosting, Extreme Gradient Boosting, etc. Along with this transfer learning was also used for the classification of pistachios.

Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task. Transfer learning involves using pre-trained neural network models that have acquired extensive knowledge from large datasets and fine-tuning them for specific tasks. This approach reduces the need for massive datasets and computational resources while benefiting from the generalization capabilities of the pre-trained models.

2.6 Features of Proposed System

• **Enhanced Accuracy:** Transfer learning significantly improves classification accuracy by utilizing pre-trained models' knowledge.

• **Improved Generalization:** The pre-trained models' ability to work effectively across different pistachio varieties shows enhanced generalization capabilities when applied to new, domain-specific tasks.

• **Robustness:** Transfer learning ensures reliable performance across different conditions and variations in pistachio images, reducing classification mistakes and improving model stability.

• **Reduced Training Time:** By leveraging pre-trained neural network models, transfer learning significantly reduces the computational resources and time required for model training.

• **Leveraging Knowledge:** Transfer learning effectively utilizes knowledge gained from pre-trained models trained on large datasets to perform well on smaller, specific datasets like pistachio classification.

2.7 Components of Proposed System

1. Input Module:
   - Image Data
     - ➢ Handles pistachio images for classification
   - Numerical Data
     - ➢ Processes 16-feature numerical data
     - ➢ Processes 28-feature numerical data
2. Processing Module:
   - Data Preprocessing
     - ➢ Feature scaling and normalization
   - Model selection and customization
     - ➢ ML algorithms selection
     - ➢ Transfer Learning models integration
     - ➢ Pre-trained model weights management
     - ➢ Classification head customization
3. Classification Module:
   - FastAI Models training
     - ➢ ResNet, VGG, DenseNet implementation
     - ➢ AlexNet and SqueezeNet implementation
   - Timm Models training
     - ➢ EfficientNet, RegNetX implementation
     - ➢ Xception model implementatio
4. Output Module:
   - Results

    - ➢ Classification predictions
    - ➢ Confidence scores
    - ➢ Performance metrics
  - Result Visualization
    - ➢ Confusion matrix
    - ➢ Performance graphs
    - ➢ Result analysis
    - ➢ Comparisons across various models

5. Web Interface Module:
   - User Input
     - ➢ Image upload functionality
     - ➢ Numerical input interface
   - Result Display
     - Output classification prediction

2.8 Selection of Hardware and Algorithms

- Hardware and Software Specifications
  - ➢ GPU – Tesla P100-PCIE
  - ➢ CPU – Intel(R) Xeon(R) @ 2.00GHz
  - ➢ Operating System – Linux 5.15.154+
  - ➢ Python – 3.10.13.final.0 (64 bit)
  - ➢ CPU RAM – 32 GB
  - ➢ GPU RAM – 16 GB
- Libraries used
  - ➢ Pandas
  - ➢ Numpy
  - ➢ Matplotlib
  - ➢ Scikit–Learn
  - ➢ Fast.ai
  - ➢ Timm
  - ➢ Streamlit

CHAPTER 3

SYSTEM DESIGN

3.1 Methodology

Methodology is a series of steps that visualizes the flow of processes in a project. It helps in efficiently utilizing resources needed to accomplish the tasks. Fig. 3.1 shows the workflow methodology of our project.

1. Data Collection
2. Exploratory Data Analysis (EDA)
3. Data Pre-processing
4. Training the models
5. Evaluating the test data
6. Result Analysis
7. Web-Application

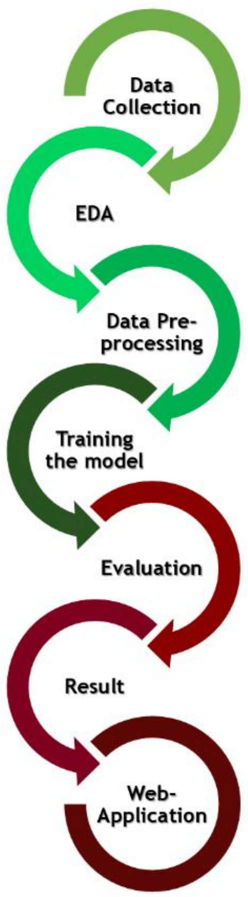


Fig. 3.1 Methodology

6.1 Data Collection

The first part involves data collection, where dataset was used from Kaggle [13]. The dataset includes a total of 2148 instances. The dataset consists of 1232 instances of Kirmizi and 916 instances of Siirt Pistachio. Dataset consists of 3 directories:

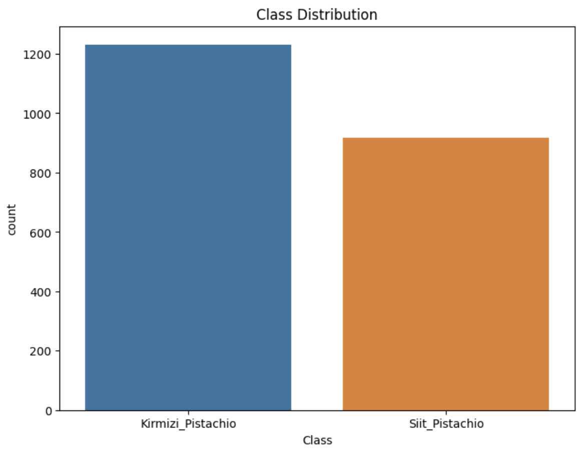
1. Numerical Data (16 features) – Excel
2. Numerical Data (28 features) – Excel

3. Image Data

6.2 Exploratory Data Analysis

EDA was performed to understand the underlying patterns, relationships, and characteristics of the pistachio dataset.

- Class Distribution Analysis – Fig 3.2 shows class distribution and Fig. 3.3 shows sample of the types of pistachios



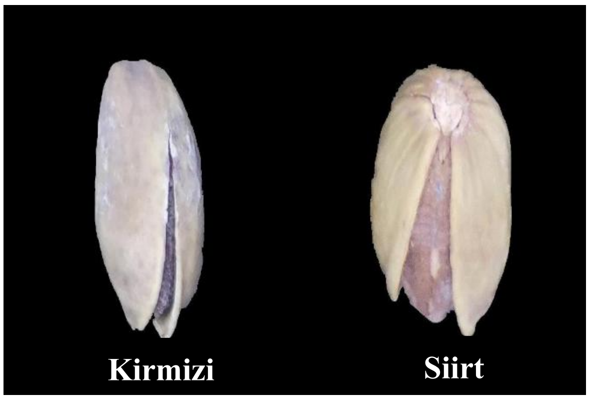


Fig. 3.2 Class Distribution of Pistachios Fig. 3.3 Types of Pistachios

- Feature Analysis of Numerical Data – 16 Features

Table 3.1 Feature Analysis – 16 Features

Morphological Features	Shape Features
1. Area	A. Shape Factor – 1
2. Perimeter	B. Shape Factor – 2
3. Major Axis	C. Shape Factor – 3
4. Minor Axis	D. Shape Factor – 4
5. Eccentricity	
6. Eqdiasq	
7. Solidity	
8. Convex Area	
9. Extent	
10. Aspect Ratio	
11. Roundness	
12. Compactness	

Table 3.1 shows the 16 features of numerical data. Here, the shape factors represent the following:

- Shape Factor – 1: Ratio of Major Axis to Area
- Shape Factor – 2: Ratio of Minor Axis to Area
- Shape Factor – 3: Ratio of Minor Axis to Major Axis
- Shape Factor – 4: Ratio of Extent to Roundness

- Feature Analysis of Numerical Data – 28 Features

Table 3.2 Feature Analysis – 28 Features

Morphological Features	**Shape Features**	**Color Features**
1. Area	1. Shape Factor – 1	1. Mean_RR
2. Perimeter	2. Shape Factor – 2	2. Mean_RG
3. Major Axis	3. Shape Factor – 3	3. Mean_RB
4. Minor Axis	4. Shape Factor – 4	4. StdDev_RR
5. Eccentricity		5. StdDev_RG
6. Eqdiasq		6. StdDev_RB
7. Solidity		7. Skew_RR
8. Convex Area		8. Skew_RG
9. Extent		9. Skew_RB
10. Aspect Ratio		10. Kurtosis_RR
11. Roundness		11. Kurtosis_RG
12. Compactness		12. Kurtosis_RB

Table 3.2 shows the 28 features of the numerical data

- Data Visualization of Numerical Data
  - Histogram: For understanding feature distributions. Fig. 3.4 and Fig. 3.5 shows the histogram of the 16 and 28 features.

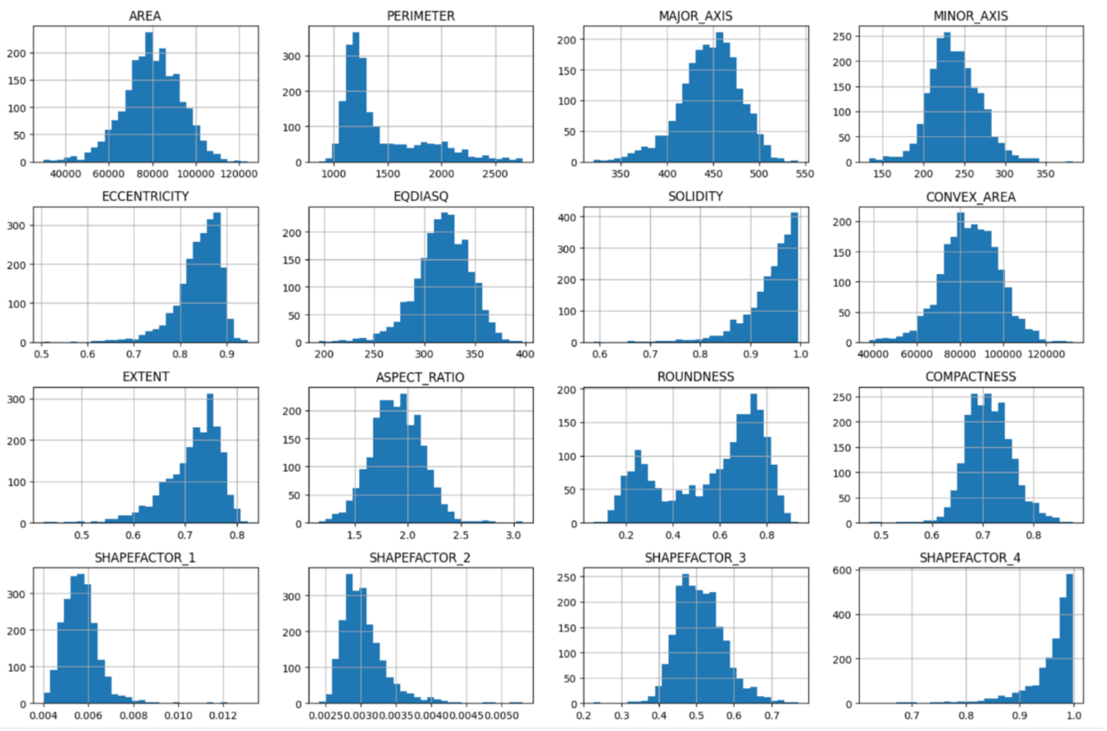


Fig. 3.4 Histogram of 16 features

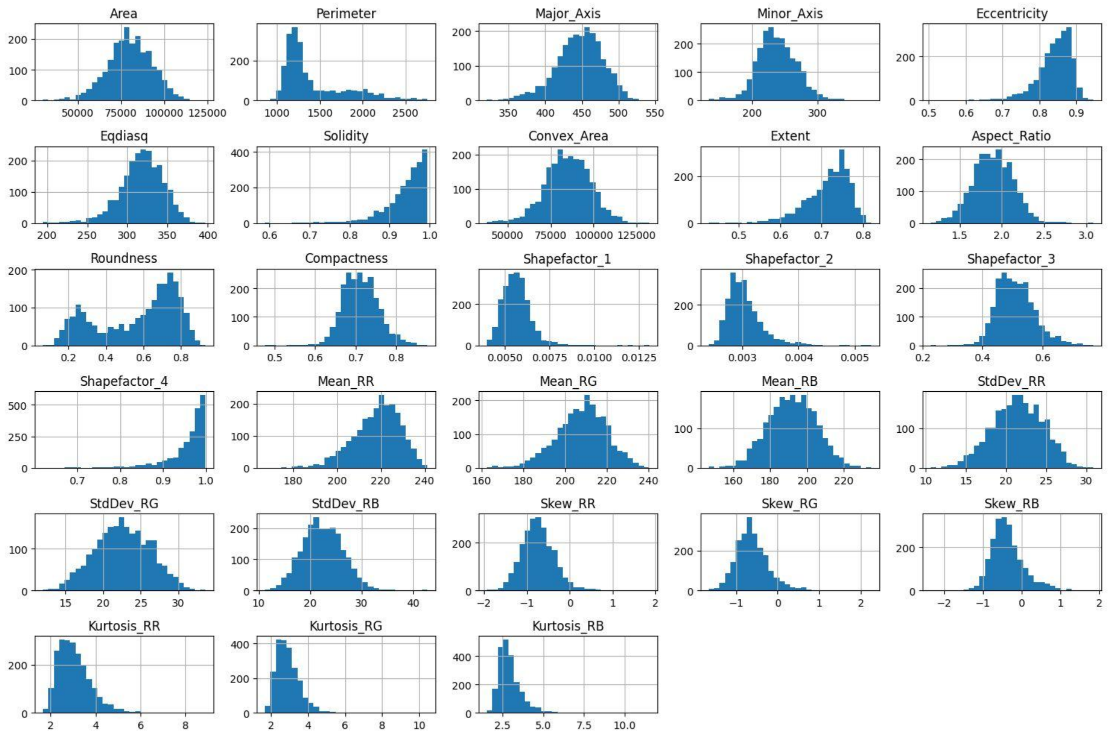


Fig. 3.5 Histogram of 28 features

- Box Plot: For detecting outliers. Fig. 3.6 and Fig. 3.7 shows the box plots of the 16 and 28 features.

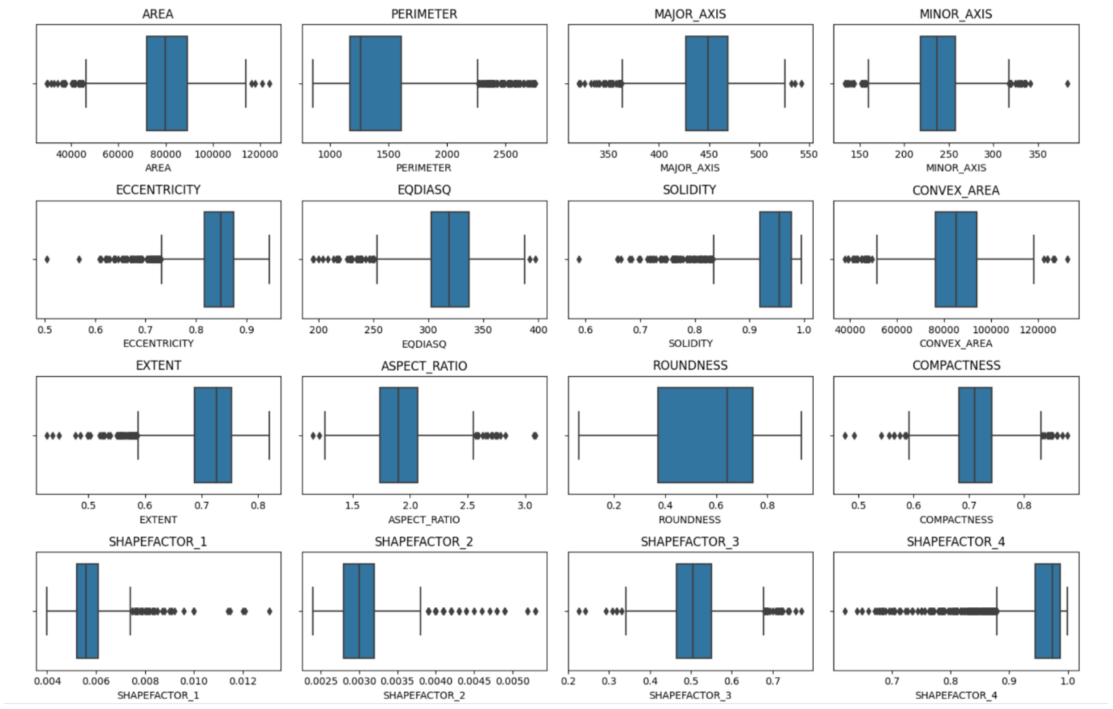


Fig. 3.6 Outliers detection using Box Plot for 16 features

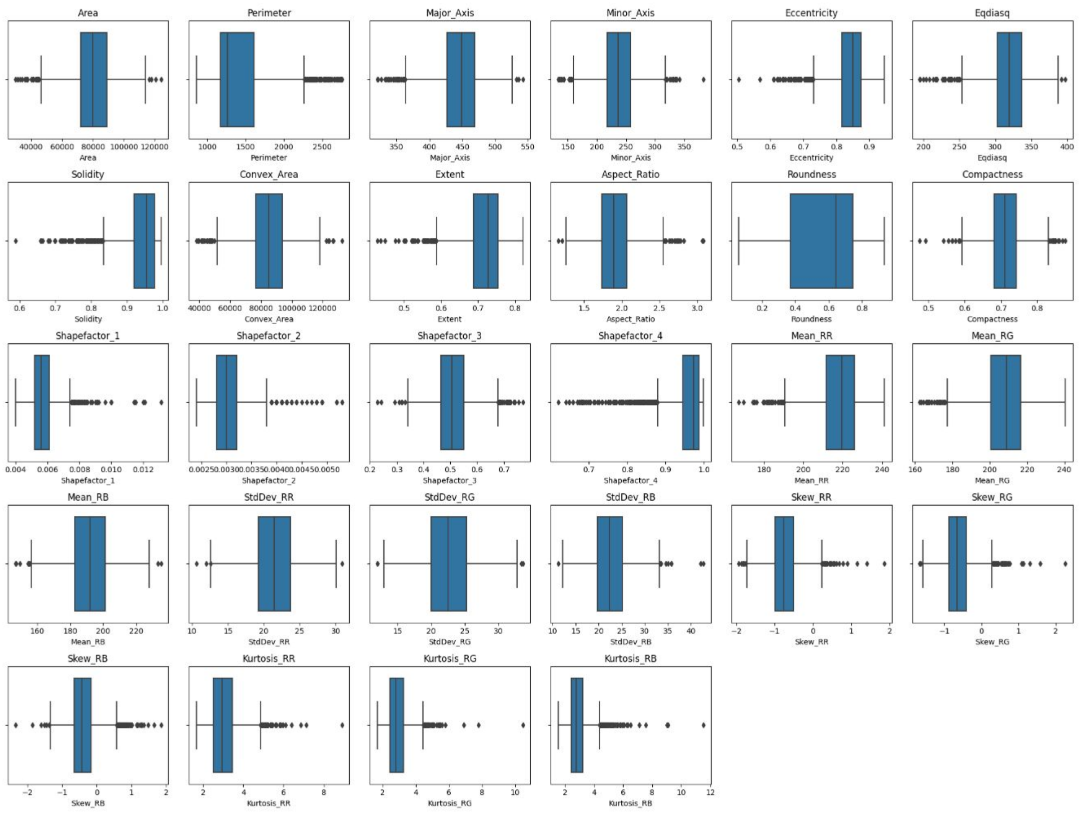


Fig. 3.7 Outliers detection using Box Plot for 28 features

- Correlation Matrix: To determine how features are correlated with each other.

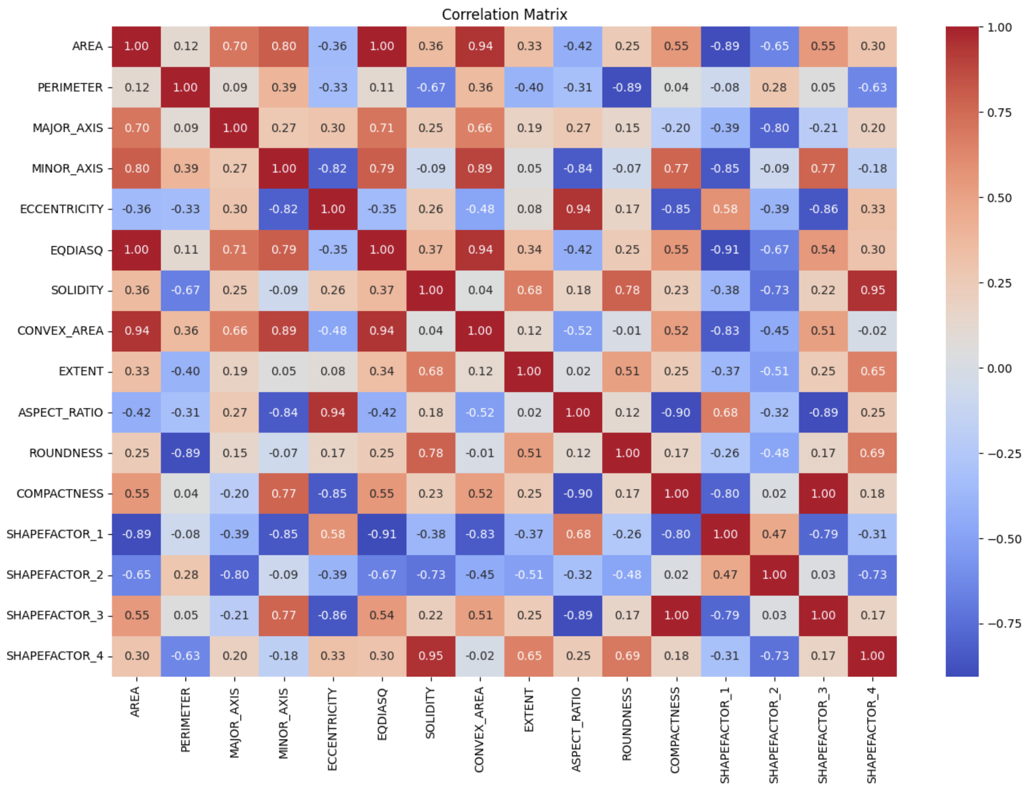


Fig. 3.8 Correlation Matrix of 16 features

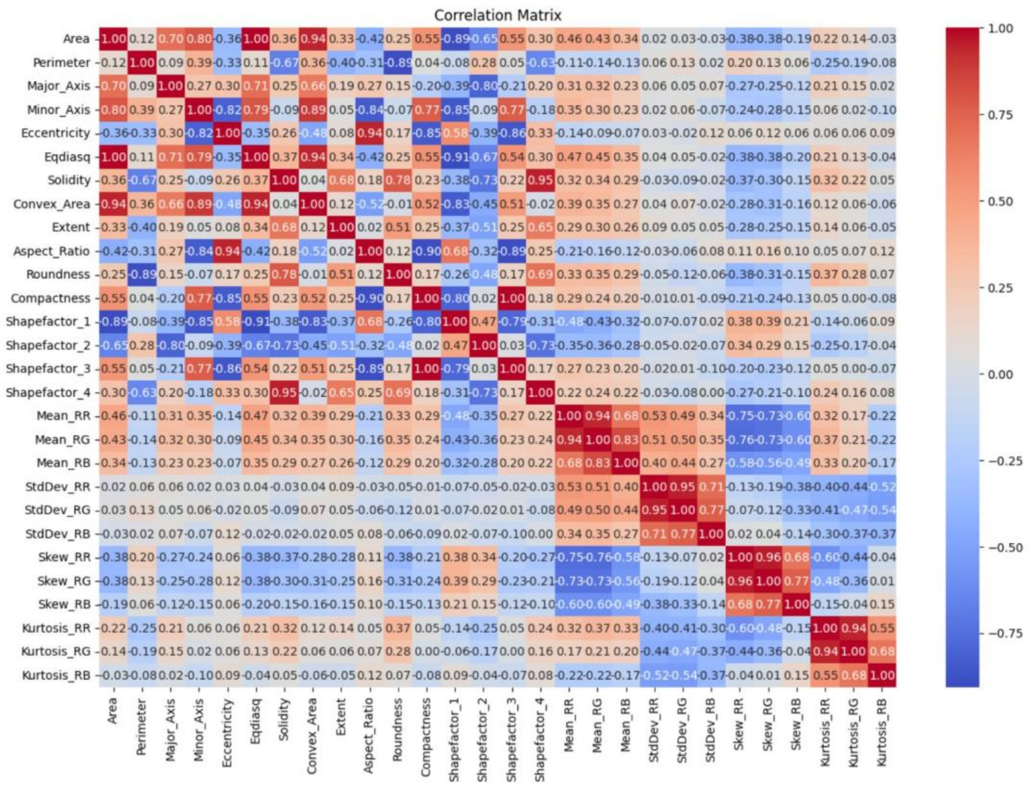


Fig. 3.9 Correlation Matrix of 28 features

Fig. 3.8 and Fig. 3.9 shows the correlation matrix of 16 and 28 features.

- Box Plots: For visualizing the distribution of each feature grouped by class.

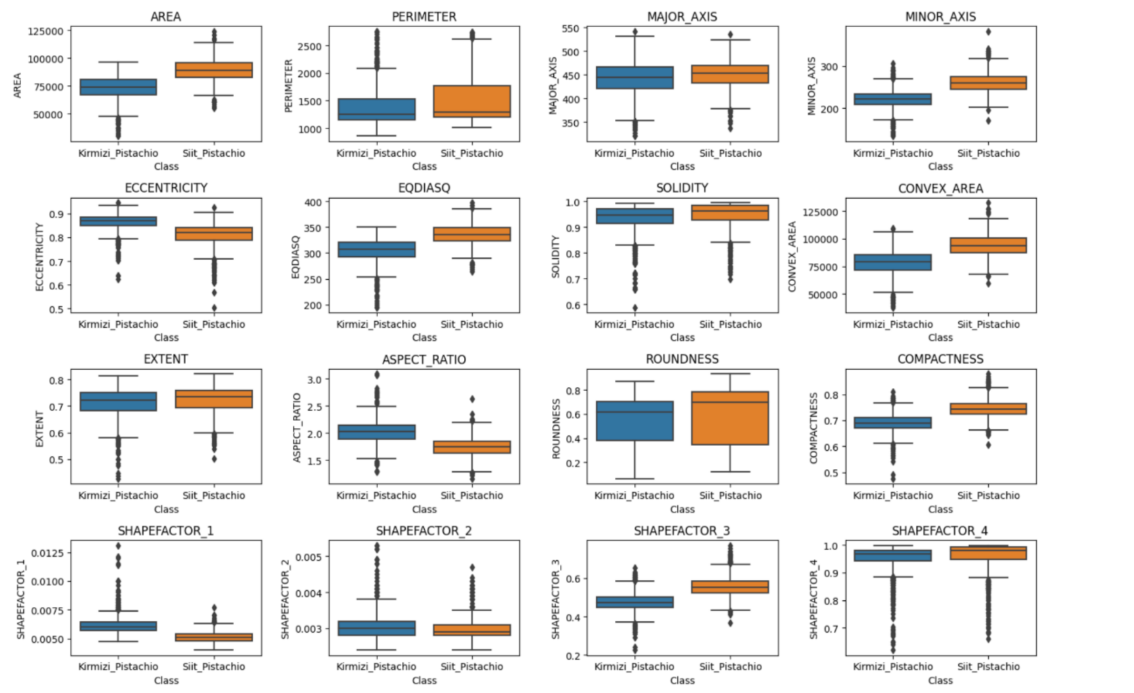


Fig. 3.10 Box Plot of 16 features

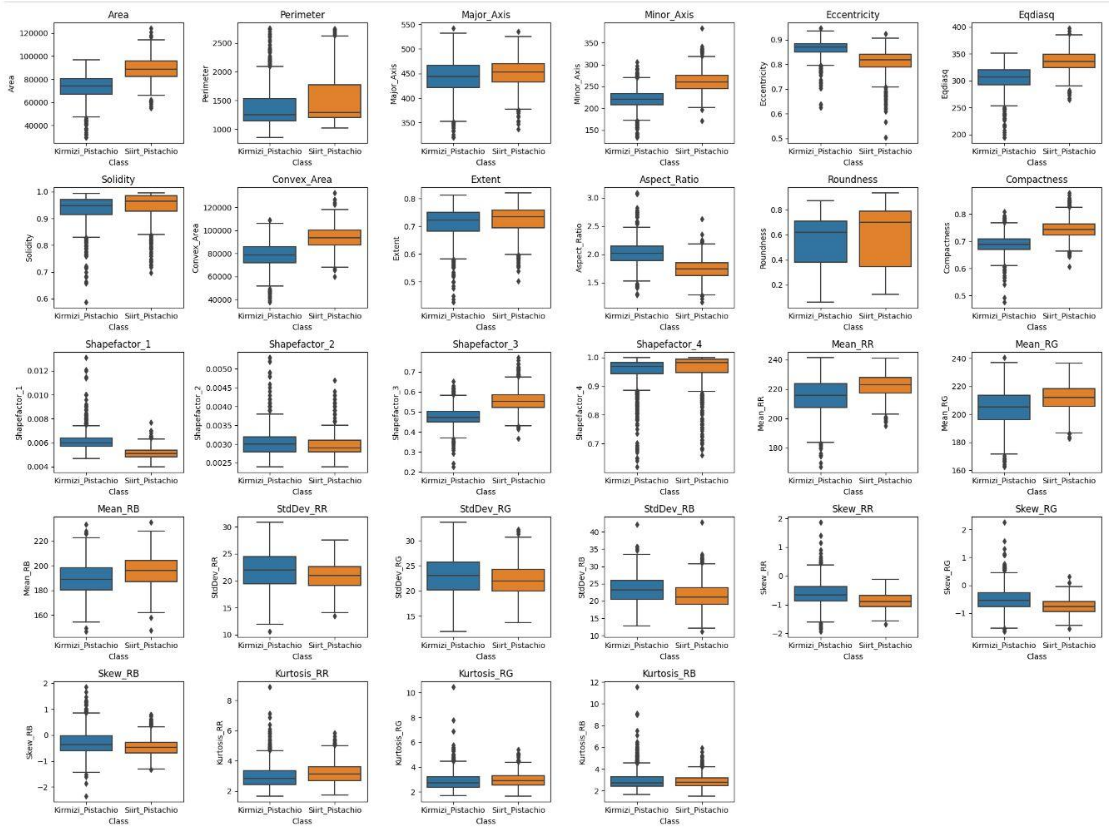


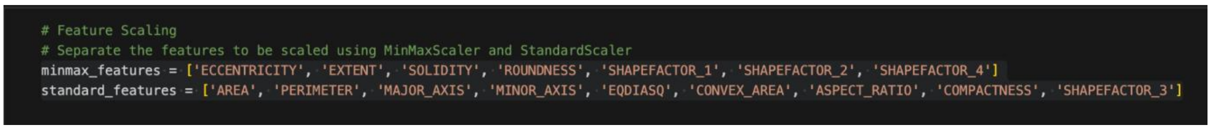
Fig. 3.11 Box Plot of 28 features

Fig. 3.10 and Fig. 3.11 shows the class distribution box plots of 16 and 28 features.

6.3 Data Pre-processing

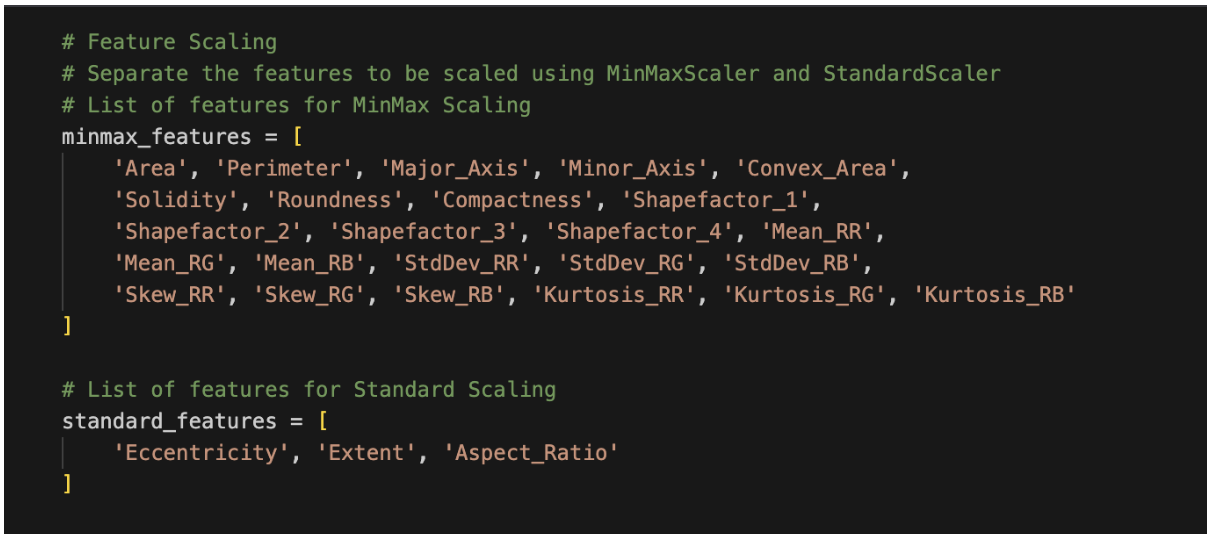
Data pre-processing was implemented to prepare the dataset for model training, ensuring optimal performance as shown in Fig. 3.12 and Fig. 3.13. The following techniques were applied:

1. Mini–Max Scaling: This normalizes the data to a fixed range, usually between 0 and 1. It is useful when the data does not follow a Gaussian distribution.
2. Standard Scaling: Also known as Z-score normalization, this transforms the data to have a mean of 0 and a standard deviation of 1. It is particularly useful for algorithms that assume normally distributed data.



```
# Feature Scaling
# Separate the features to be scaled using MinMaxScaler and StandardScaler
minmax_features = ['ECCENTRICITY', 'EXTENT', 'SOLIDITY', 'ROUNDNESS', 'SHAPEFACTOR_1', 'SHAPEFACTOR_2', 'SHAPEFACTOR_4']
standard_features = ['AREA', 'PERIMETER', 'MAJOR_AXIS', 'MINOR_AXIS', 'EQDIASQ', 'CONVEX_AREA', 'ASPECT_RATIO', 'COMPACTNESS', 'SHAPEFACTOR_3']
```

Fig. 3.12 Preprocessing for 16 features



```
# Feature Scaling
# Separate the features to be scaled using MinMaxScaler and StandardScaler
# List of features for MinMax Scaling
minmax_features = [
    'Area', 'Perimeter', 'Major_Axis', 'Minor_Axis', 'Convex_Area',
    'Solidity', 'Roundness', 'Compactness', 'Shapefactor_1',
    'Shapefactor_2', 'Shapefactor_3', 'Shapefactor_4', 'Mean_RR',
    'Mean_RG', 'Mean_RB', 'StdDev_RR', 'StdDev_RG', 'StdDev_RB',
    'Skew_RR', 'Skew_RG', 'Skew_RB', 'Kurtosis_RR', 'Kurtosis_RG', 'Kurtosis_RB'
]

# List of features for Standard Scaling
standard_features = [
    'Eccentricity', 'Extent', 'Aspect_Ratio'
]
```

Fig. 3.13 Preprocessing for 28 features

6.4 Training of the model

- Training for Machine Learning Algorithms

  For the numerical data 13 Machine Learning Algorithms were applied which are as follows:

  - Basic models: Logistic Regression, KNN, Decision Tree, Naïve Bayes, SVM – Linear Kernel
  - Ensemble models: AdaBoost, CatBoost, Gradient Boosting, Extreme Gradient Boosting, Random Forest, Light Gradient Boosting, Extra Trees Classifier
  - Advanced models: Quadratic Discriminant Analysis

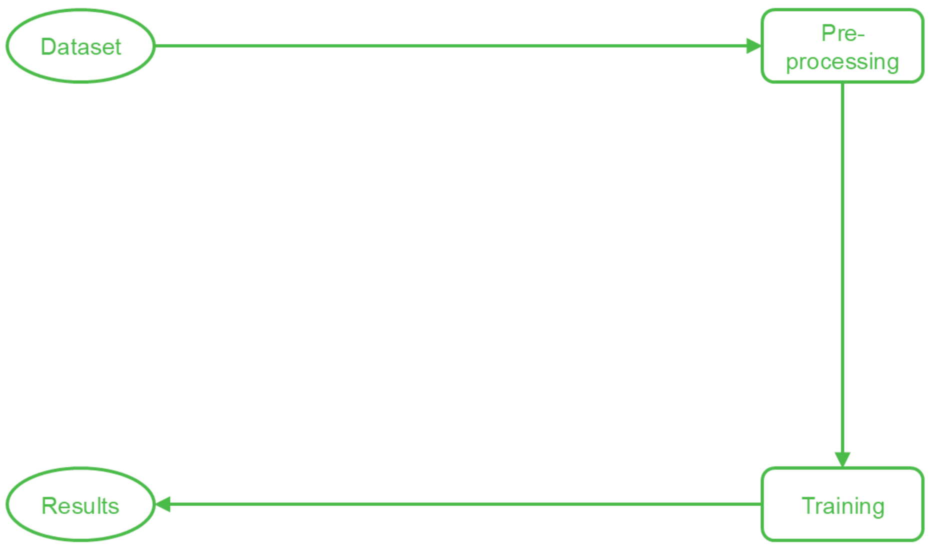


Fig. 3.14 Training stages for ML algorithms on Numerical Data

- Training for Transfer Learning Models
  - ❖ Steps for implementing Transfer Learning
    - ➢ Selected a pre-trained model
    - ➢ Loaded the model with image dataset
    - ➢ Replace the classification head (This will be a layer with 2 output nodes corresponding to the two pistachio classes: Kirmizi and Siirt).
    - ➢ Freeze the weights of the initial layers of the pre-trained model so they do not change during training.
    - ➢ Train the Model
    - ➢ Evaluate the results

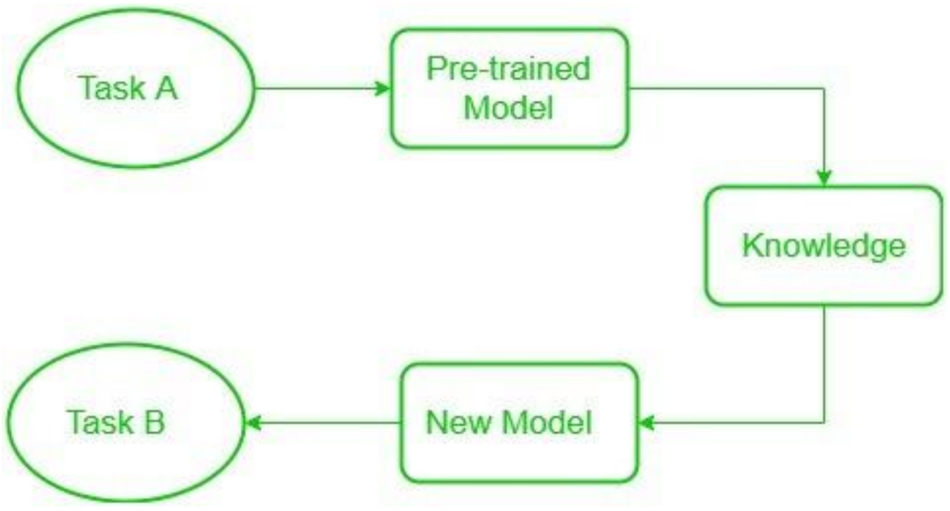


Fig. 3.15 Training stages for Image Data

Transfer Learning was implemented using two main libraries Timm and Fast.ai. Training was done by considering the batch size of 32×32.

- ❖ Fast.ai
  - ➢ Fast.ai is a deep learning library built on top of PyTorch that aims to make the power of deep learning accessible to everyone.
  - ➢ Fast.ai makes it easy to build advanced machine learning models with just a few lines of code.
  - ➢ It has tools that quickly create powerful models without needing to understand all the technical details.
  - ➢ The models used are – ResNet (5 variants), VGG (4 variants), DenseNet (4 variants), SqueezeNet (2 variants) and AlexNet.
- ❖ Timm
  - ➢ **timm** stands for **PyTorch Image Models**, which is a library developed by **Ross Wightman**.
  - ➢ It provides a large collection of pre-trained deep learning models for image classification, object detection, and other computer vision tasks.
  - ➢ The models used are – EfficientNet (6 variants), RegNetX (12 variants), and Xception.
  - ➢ Features:
    1. Wide variety of models
    2. Pre-trained weights
    3. Efficient architecture
    4. Flexible interface
    5. Data augmentation support

6.5 Evaluating the test data

The models were then used for predictions on unseen test data and results were calculated for all the algorithms.

6.6 Result Analysis

Evaluating the model's performance on unseen data using metrics helps to analyse the model's strengths and weaknesses based on the evaluation results and iteratively refine the training process or model architecture for better performance. Below are the evaluation metrics used to examine the model performance.

1. Confusion Matrix:

   A confusion matrix is a tabular representation of actual vs. predicted classes. It consists of four categories: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These values can be calculated by comparing the ground truth labels with the predicted labels.
2. Accuracy:

   Accuracy is used to measure the overall performance of the model. It is calculated as *(TP + TN) / (TP + FP + TN + FN)*.
3. Precision:

   Precision measures the proportion of true positive detections among all positive predictions. It is calculated as *TP / (TP + FP)*.
4. Recall:

   Recall (also known as sensitivity) measures the proportion of true positive detections among all actual positives. It is calculated as *TP / (TP + FN)*.
5. F1 Score:

   The F1 Score is a harmonic mean of precision and recall, providing a balance of both metrics. It is calculated as *2 × (Precision × Recall) / (Precision + Recall)*.
6. AUC ROC:

   The AUC ROC curve is a performance measurement for classification models that shows the model's ability to distinguish between the pistachio classes across various threshold settings. A higher AUC value, closer to 1.0, indicates better model performance in distinguishing between the classes.
7. Error Rate:

   Error Rate represents the proportion of incorrect predictions made by the model out of all predictions.

6.7 Web Application

The models were then incorporated into web-based application

3.2 Output and Interface Design

A web-application [14] using Streamlit where a user can choose for numerical inputs (industrial purposes) or image-based (general purpose) input, and get the output with predicted pistachio class and confidence score. Streamlit is a Python framework enabling quick building of interactive web applications. It simplifies interface creation by allowing Python functions to add widgets like sliders, buttons, and text inputs. It can easily construct dynamic interfaces because of its declarative syntax and vast widget library. Its easy connection with data science libraries and simple deployment choices makes it a popular choice for developing interactive data visualization and machine learning applications. Fig. 3.16, Fig. 3.17, and Fig. 3.18 shows the base wireframe of our web application . The main reasons for choosing Streamlit were:

- Simplicity and ease of use
- Rapid prototyping
- Built-in widgets and components
- Automatic layout optimization

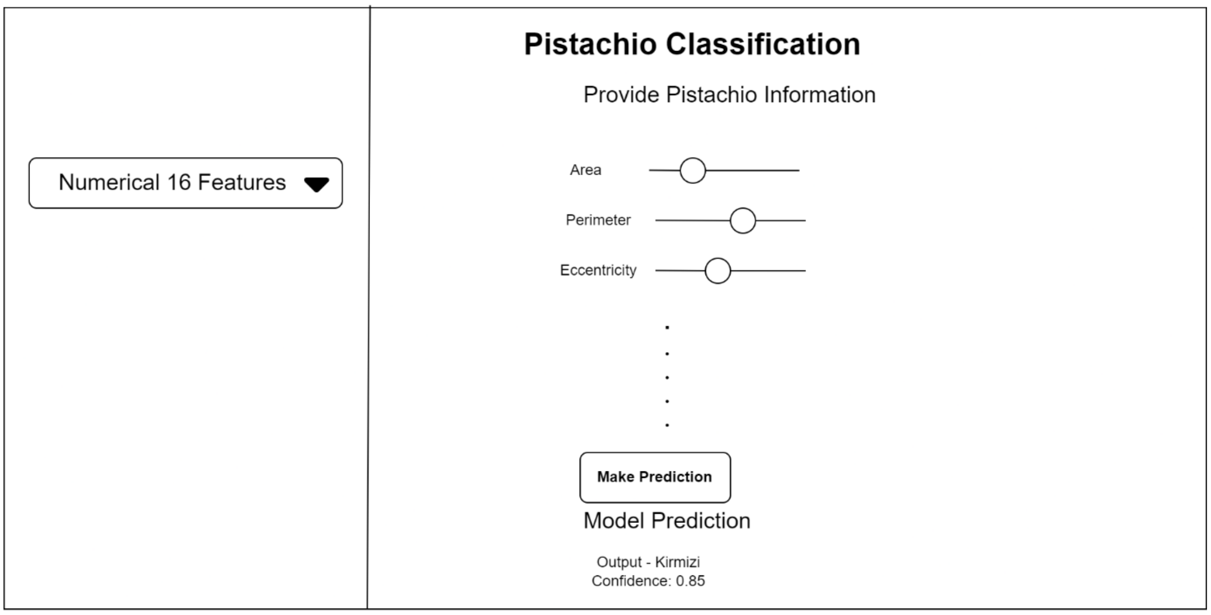


Fig. 3.16 Wireframe for numerical input – 16 features

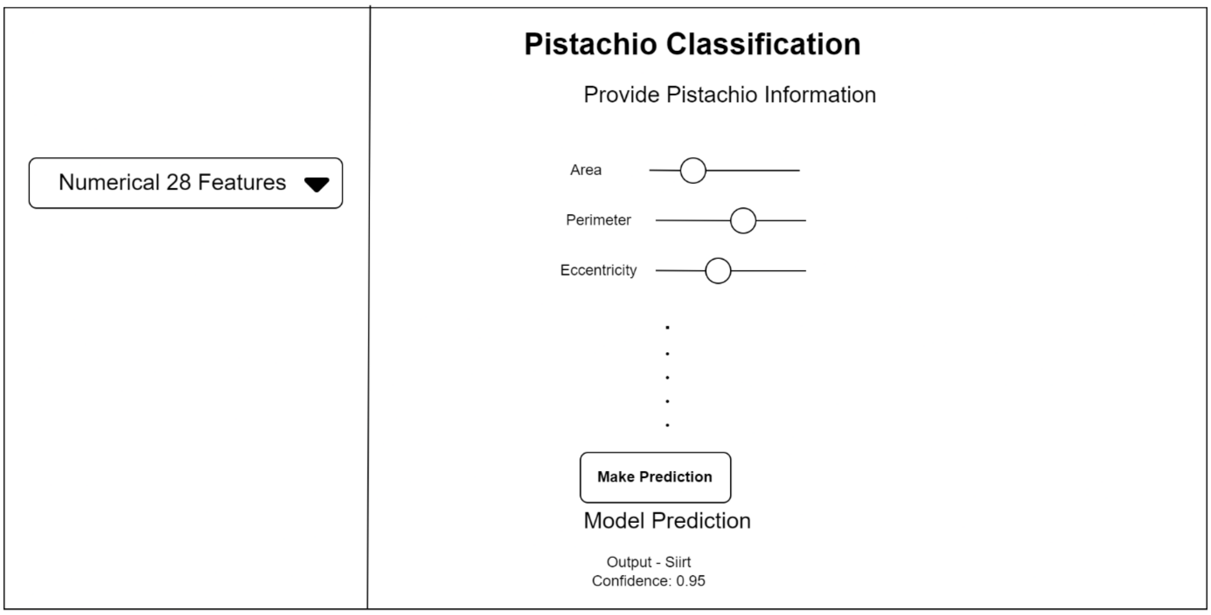


Fig. 3.17 Wireframe for numerical input – 28 features

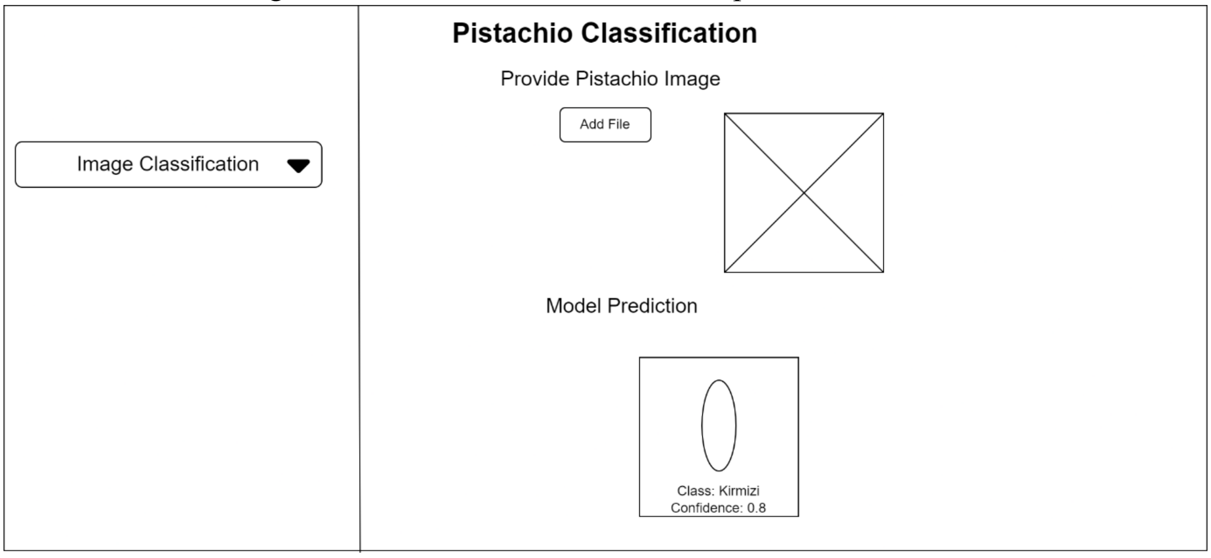


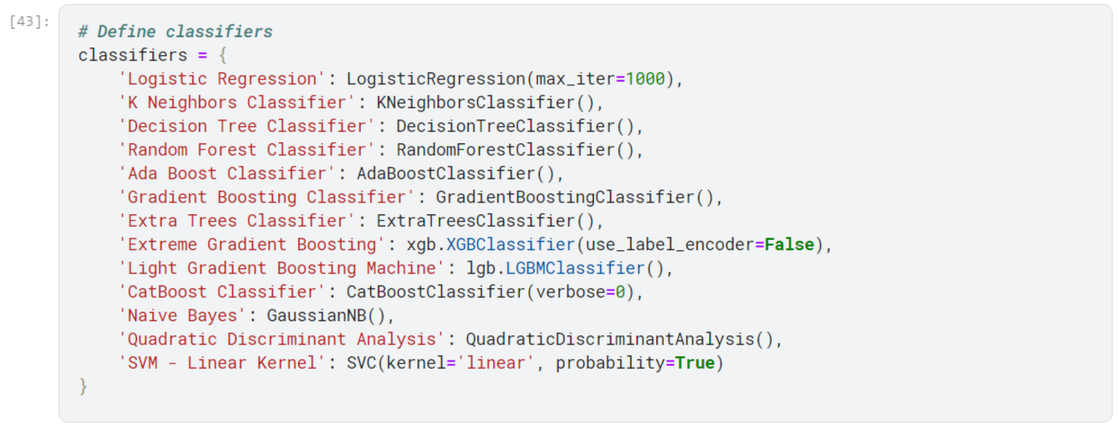
Fig. 3.18 Wireframe for image classification

CHAPTER 4

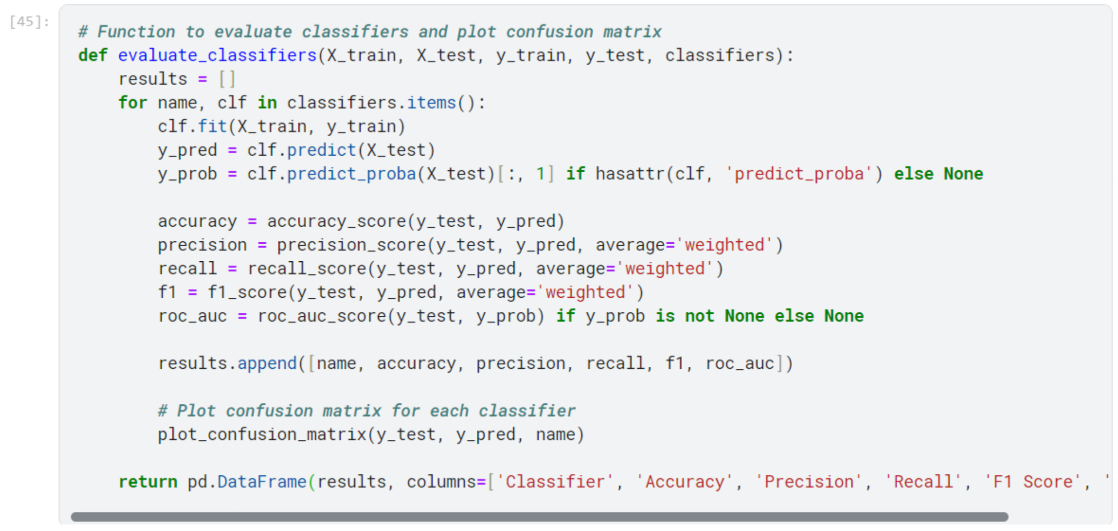
IMPLEMENTATION

4.1 Implementation

During implementation, we first trained the numerical data using the Machine Learning algorithms mentioned in previous section. We individually trained each algorithm for both numerical categories: 16 features and 28 features (shown in Fig. 4.1 for 16 features and Fig. 4.2 for 28 features). The results were calculated and are discussed in later sections.



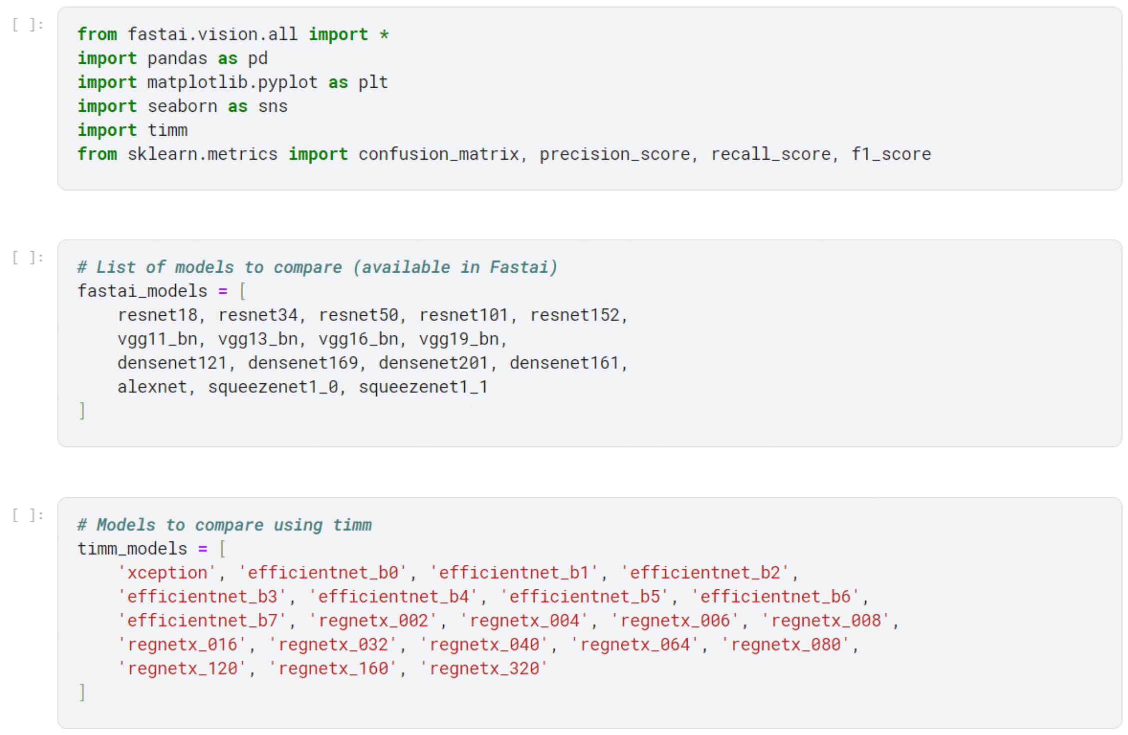
[43]:
```python
# Define classifiers
classifiers = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'K Neighbors Classifier': KNeighborsClassifier(),
    'Decision Tree Classifier': DecisionTreeClassifier(),
    'Random Forest Classifier': RandomForestClassifier(),
    'Ada Boost Classifier': AdaBoostClassifier(),
    'Gradient Boosting Classifier': GradientBoostingClassifier(),
    'Extra Trees Classifier': ExtraTreesClassifier(),
    'Extreme Gradient Boosting': xgb.XGBClassifier(use_label_encoder=False),
    'Light Gradient Boosting Machine': lgb.LGBMClassifier(),
    'CatBoost Classifier': CatBoostClassifier(verbose=0),
    'Naive Bayes': GaussianNB(),
    'Quadratic Discriminant Analysis': QuadraticDiscriminantAnalysis(),
    'SVM - Linear Kernel': SVC(kernel='linear', probability=True)
}
```



[45]:
```python
# Function to evaluate classifiers and plot confusion matrix
def evaluate_classifiers(X_train, X_test, y_train, y_test, classifiers):
    results = []
    for name, clf in classifiers.items():
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        y_prob = clf.predict_proba(X_test)[:, 1] if hasattr(clf, 'predict_proba') else None

        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        f1 = f1_score(y_test, y_pred, average='weighted')
        roc_auc = roc_auc_score(y_test, y_prob) if y_prob is not None else None

        results.append([name, accuracy, precision, recall, f1, roc_auc])

        # Plot confusion matrix for each classifier
        plot_confusion_matrix(y_test, y_pred, name)

    return pd.DataFrame(results, columns=['Classifier', 'Accuracy', 'Precision', 'Recall', 'F1 Score', '
```

Fig. 4.1 Code snippet of implementation on numerical data

For the image data, total of 35 different models were trained using both Fast.ai and Timm PyTorch libraries.



```
[ ]: from fastai.vision.all import *
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import timm
     from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
```

```
[ ]: # List of models to compare (available in Fastai)
     fastai_models = [
         resnet18, resnet34, resnet50, resnet101, resnet152,
         vgg11_bn, vgg13_bn, vgg16_bn, vgg19_bn,
         densenet121, densenet169, densenet201, densenet161,
         alexnet, squeezenet1_0, squeezenet1_1
     ]
```

```
[ ]: # Models to compare using timm
     timm_models = [
         'xception', 'efficientnet_b0', 'efficientnet_b1', 'efficientnet_b2',
         'efficientnet_b3', 'efficientnet_b4', 'efficientnet_b5', 'efficientnet_b6',
         'efficientnet_b7', 'regnetx_002', 'regnetx_004', 'regnetx_006', 'regnetx_008',
         'regnetx_016', 'regnetx_032', 'regnetx_040', 'regnetx_064', 'regnetx_080',
         'regnetx_120', 'regnetx_160', 'regnetx_320'
     ]
```

Fig. 4.2 Code snippet of implementation on image data

4.2 Results

The results contain confusion matrix and evaluation metric graphs. It is divided into four sections:

- Machine Learning Algorithms on Numerical Data – Fig. 4.3 and Fig. 4.4 shows the confusion matrix of best models from 16 feature training (Gradient Boosting) and 28 feature training (CatBoost)

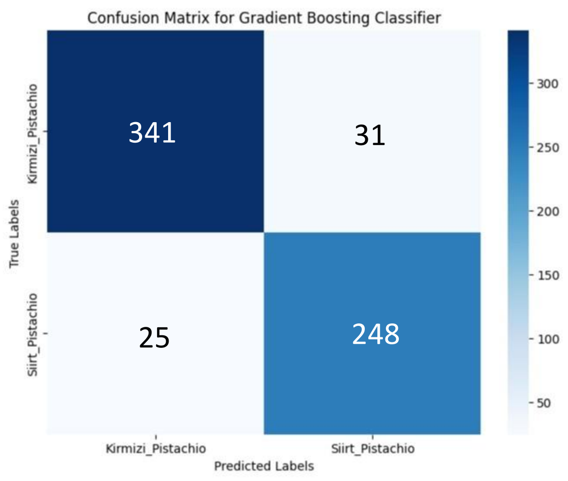


Fig. 4.3 Confusion Matrix of Best Performing Model – 16 features

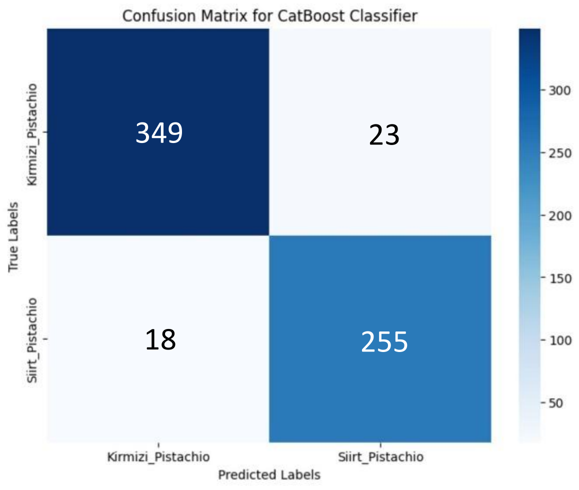


Fig. 4.4 Confusion Matrix of Best Performing Model – 28 features

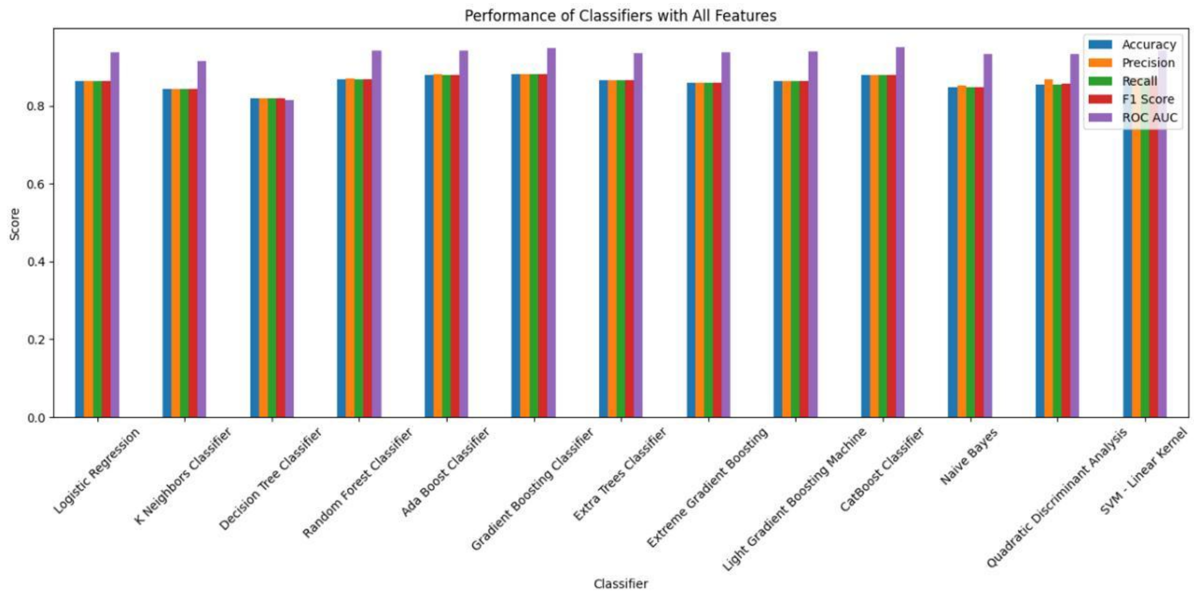


Fig. 4.5 Model Comparison – 16 Features

Fig. 4.5 shows the comparison of all the 13 algorithms of 16 features.

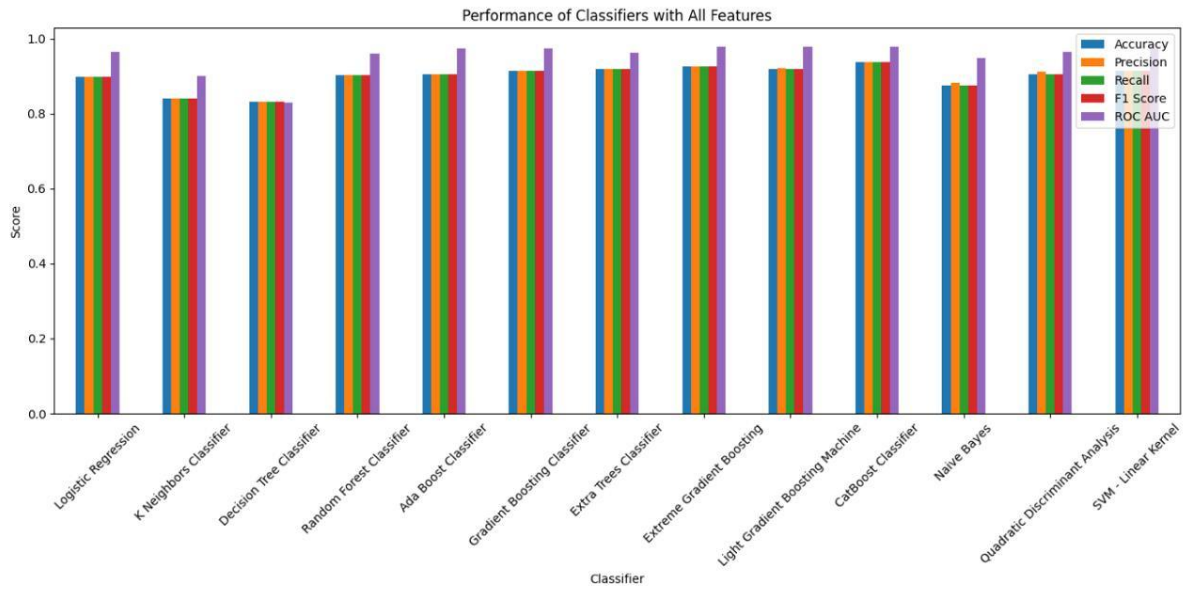


Fig. 4.6 Model Comparison – 28 Features

Fig. 4.5 shows the comparison of all the 13 algorithms of 16 features.

- Fast.ai models on Image Data

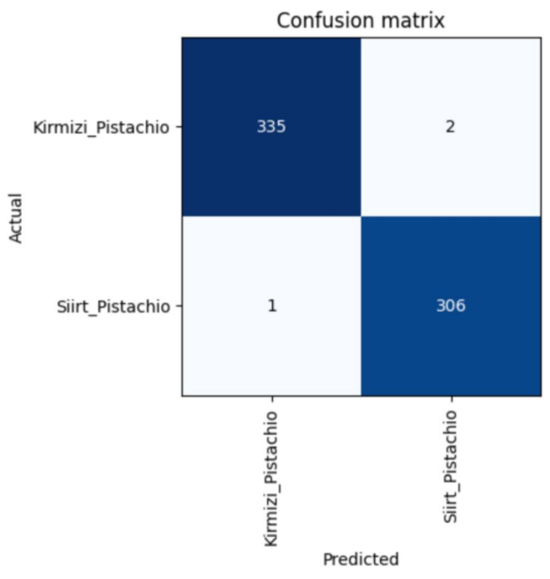


Fig. 4.7 Confusion Matrix of best variant of ResNet (ResNet34)

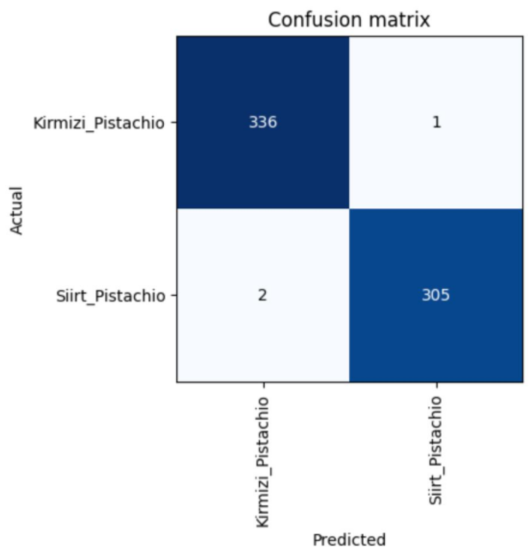


Fig. 4.8 Confusion Matrix of best variant of VGG (VGG19_bn)

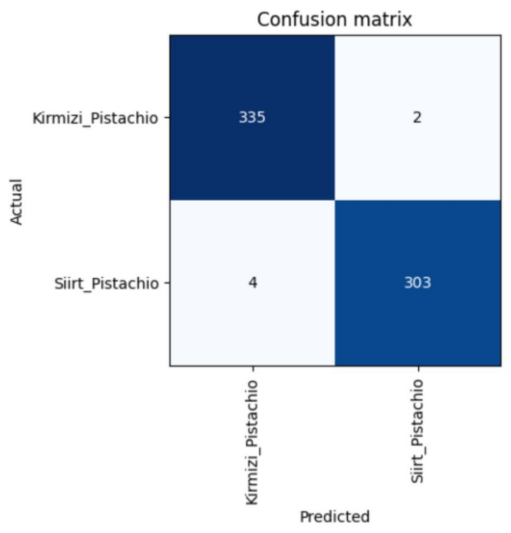


Fig. 4.9 Confusion Matrix of best variant of DenseNet (DenseNet201)

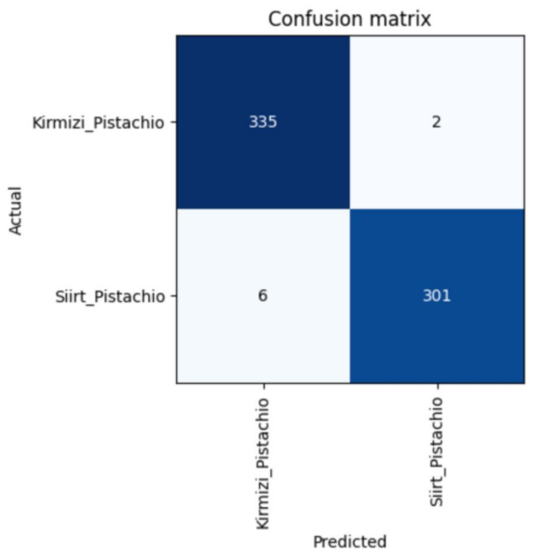


Fig. 4.10 Confusion Matrix of AlexNet

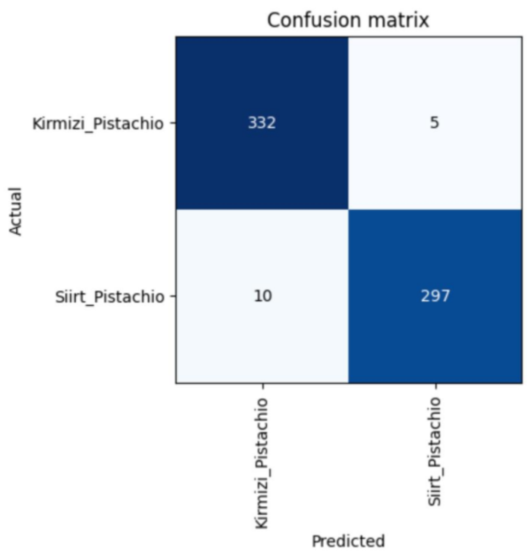


Fig. 4.11 Confusion Matrix of best variant of SqueezeNet (SqueezeNet1_1)

Fig. 4.7, 4.8, 4.9, 4.10, 4.11 shows the confusion matrix of best variants from each pre-trained model of Fast.ai library.

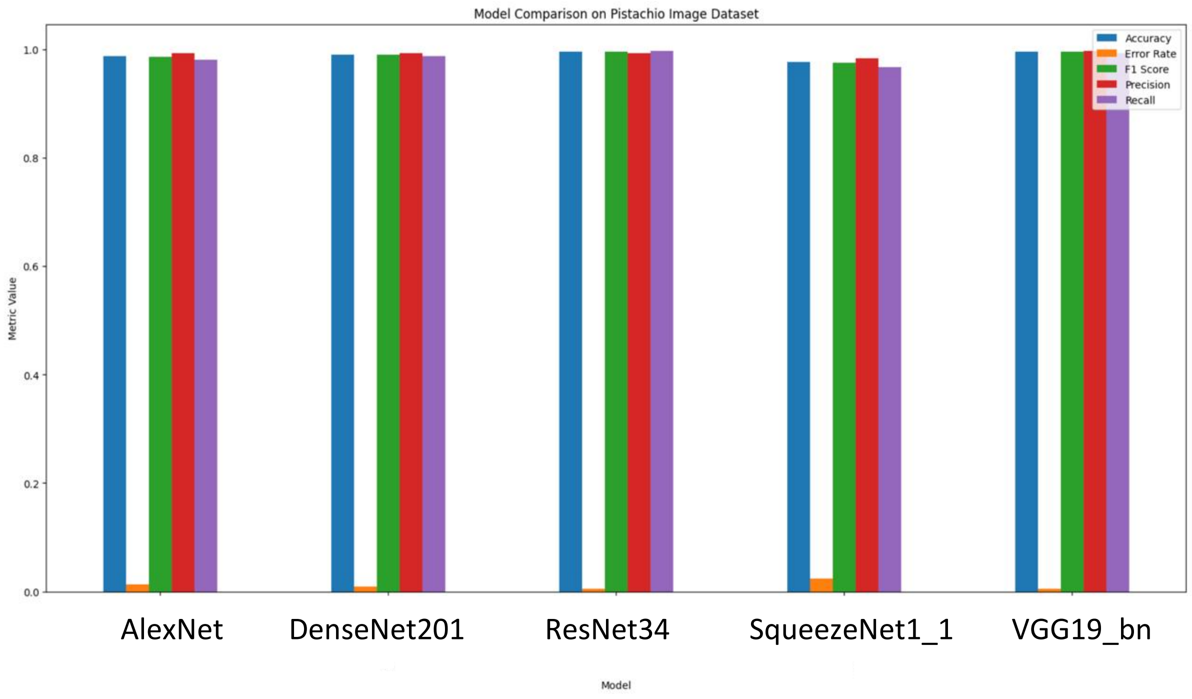


Fig. 4.12 Comparison of best performing variants of each model

Fig. 4.12 shows the comparison of the best performing models from each pre-trained model. Fig. 4.13 and Fig. 4.14 shows comparison by accuracy and error rate. Fig. 4.15, Fig. 4.16, Fig. 4.17, and Fig. 4.18 shows the comparison of best models from Timm library by accuracy, precision, recall and F1 Score.

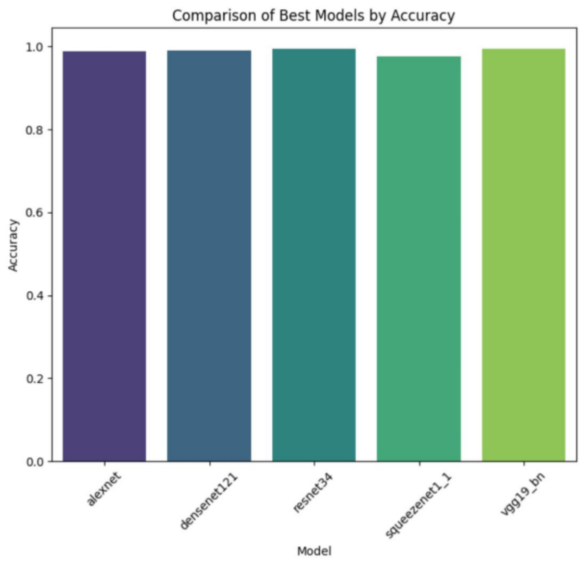


Fig. 4.13 Comparison of best performing variants of each model by accuracy

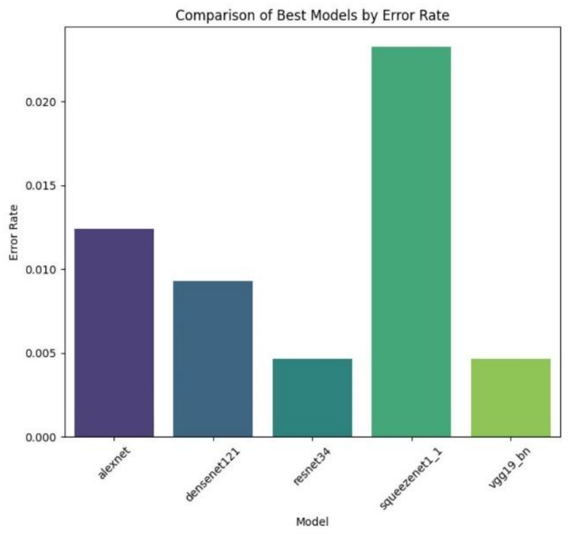


Fig. 4.14 Comparison of best performing variants of each model by error rate

- Timm models on Image Data

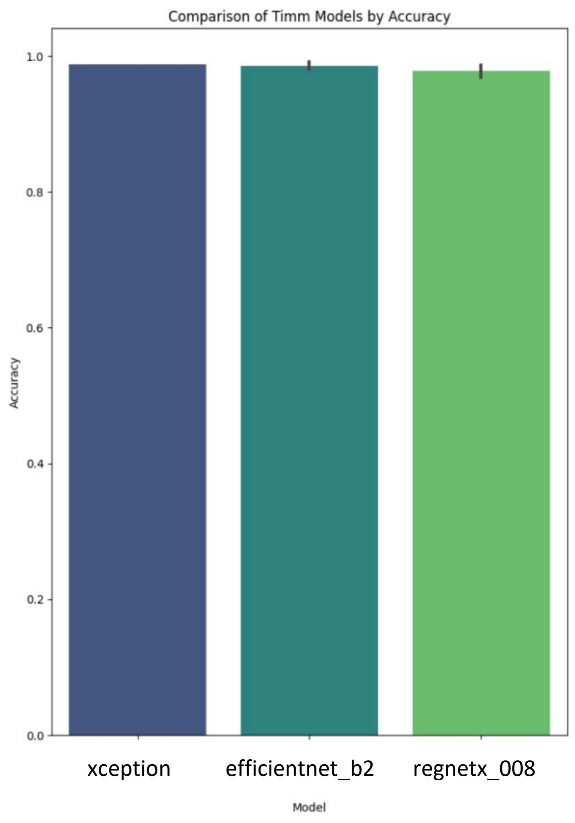


Fig. 4.15 Comparison of best performing variants of each model by accuracy

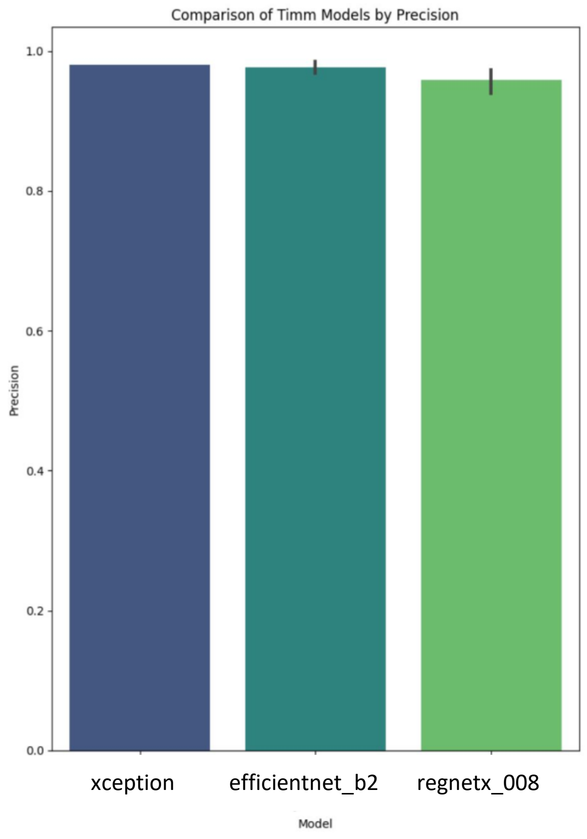
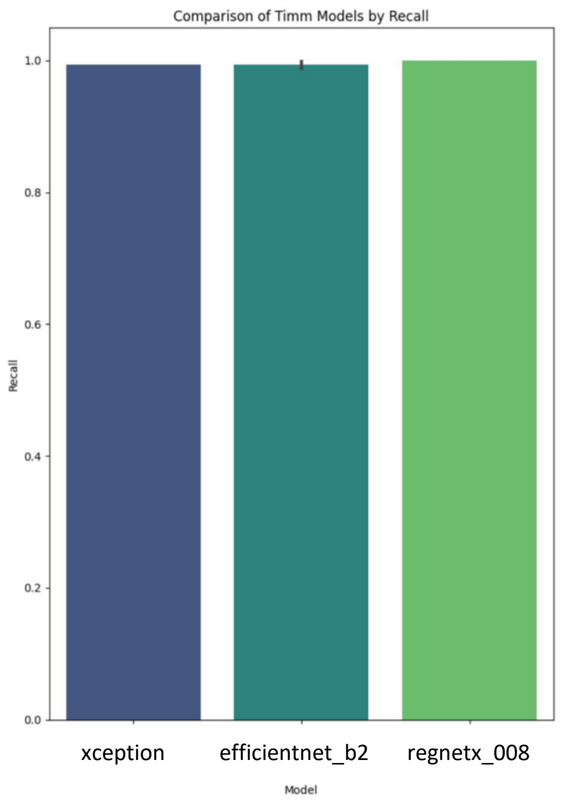


Fig. 4.16 Comparison of best performing variants of each model by precision



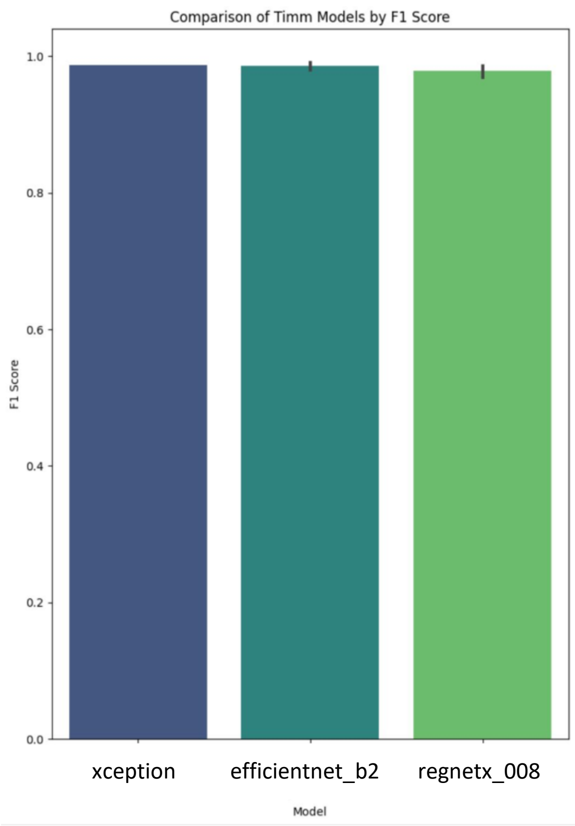


Fig. 4.17 Comparison of best performing variants of each model by recall

Fig. 4.18 Comparison of best performing variants of each model by F1 Score

- Comparative Results of Fast.ai and Timm

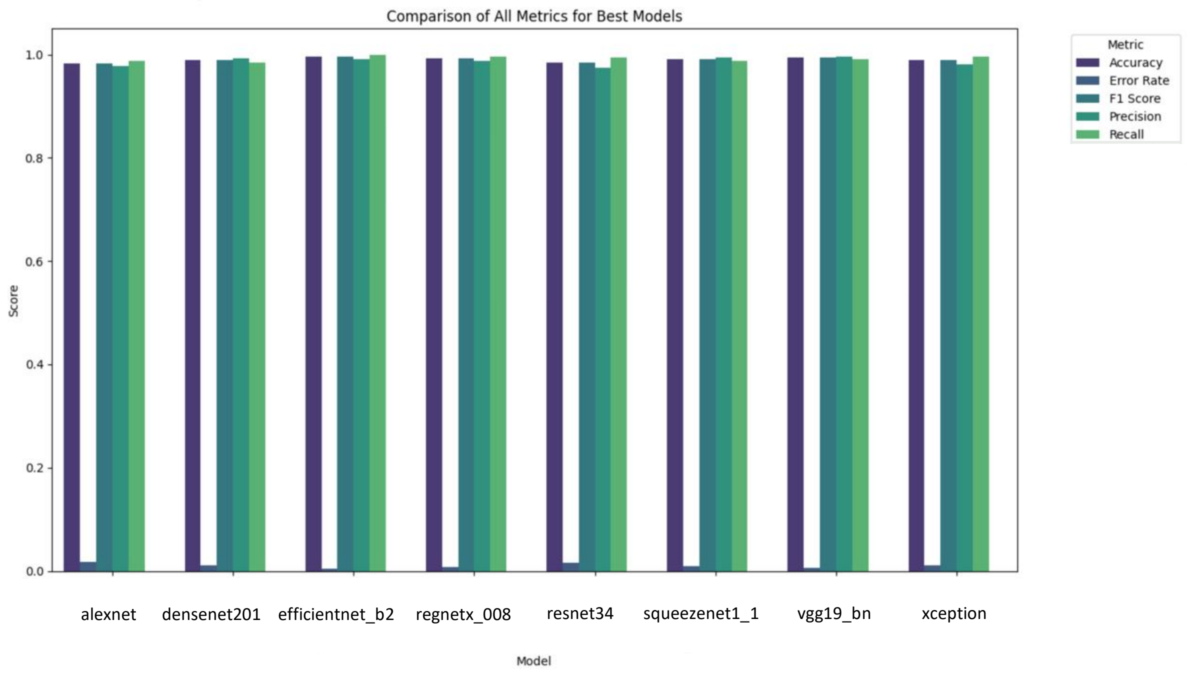


Fig. 4.19 Best performing model comparison of Fast.ai and Timm

Fig. 4.19 shows the comparison of best models from both Fast.ai and Timm library.

4.3 Output

Finally, after the model training, we feed the model with the images that model has not seen during its training. Based on model's prediction and confidence scores, we check which model is best suited for application. After finding which model is best suited, we use those models in our web application for accurate real-time pistachio classification.

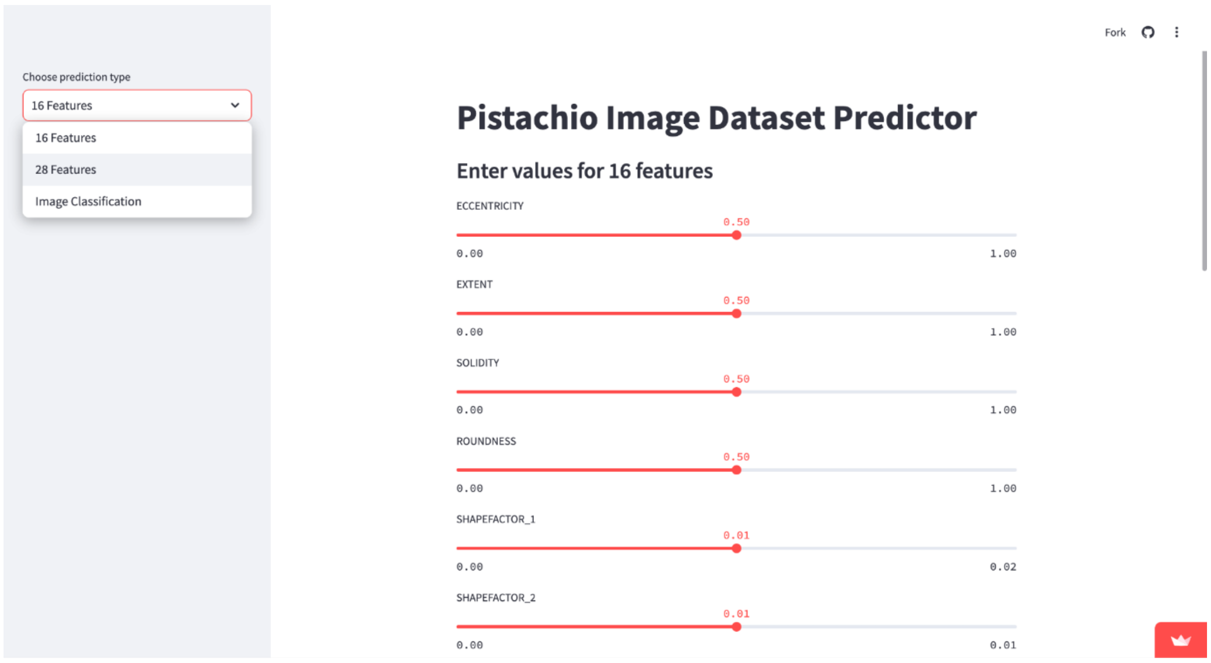


Fig. 4.20 Home page of web-application

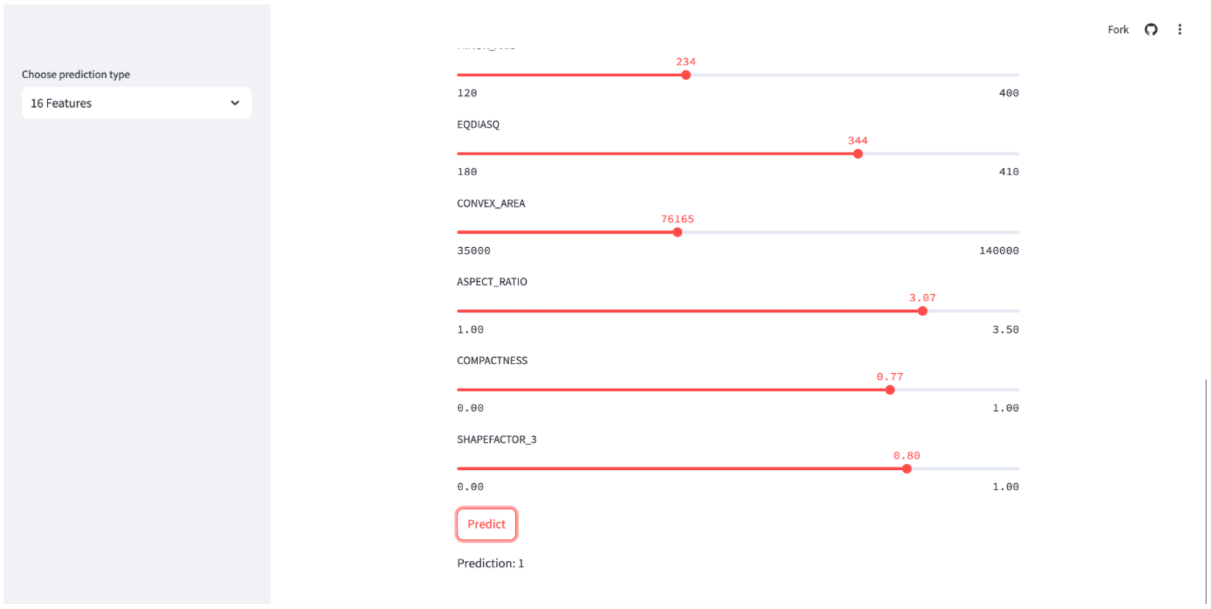
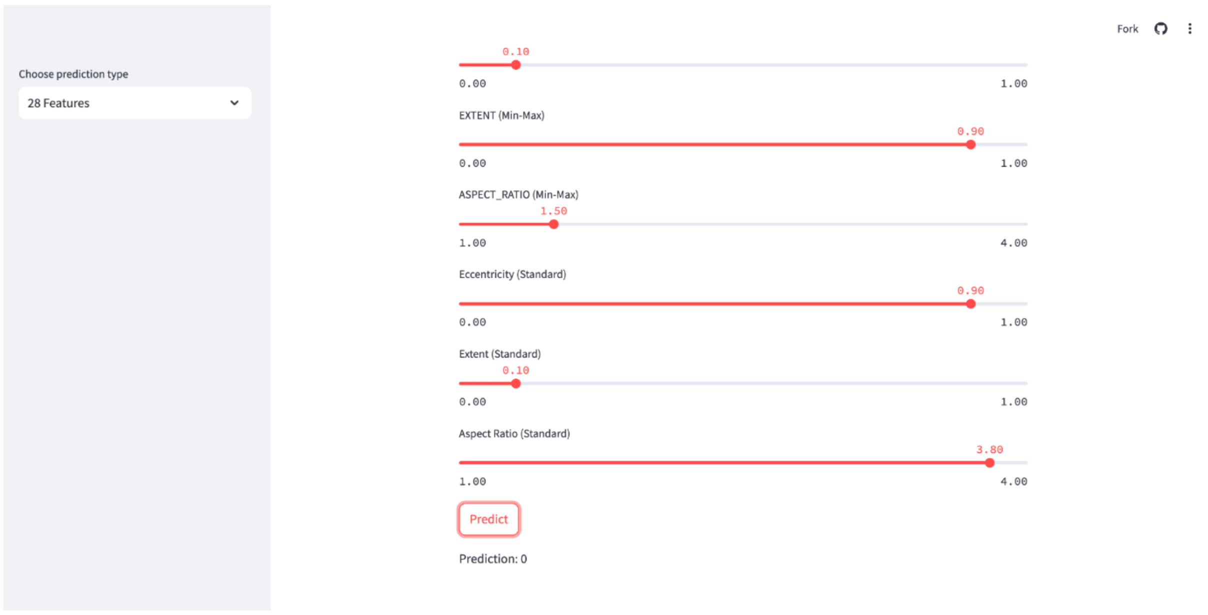


Fig. 4.21 Numerical input for 16 features

Fig. 4.20 shows the home page for the numerical feature input. Fig. 4.21 shows input page for 16 features numerical data.



Choose prediction type
28 Features

0.10
0.00 1.00

EXTENT (Min-Max)
0.90
0.00 1.00

ASPECT_RATIO (Min-Max)
1.50
1.00 4.00

Eccentricity (Standard)
0.90
0.00 1.00

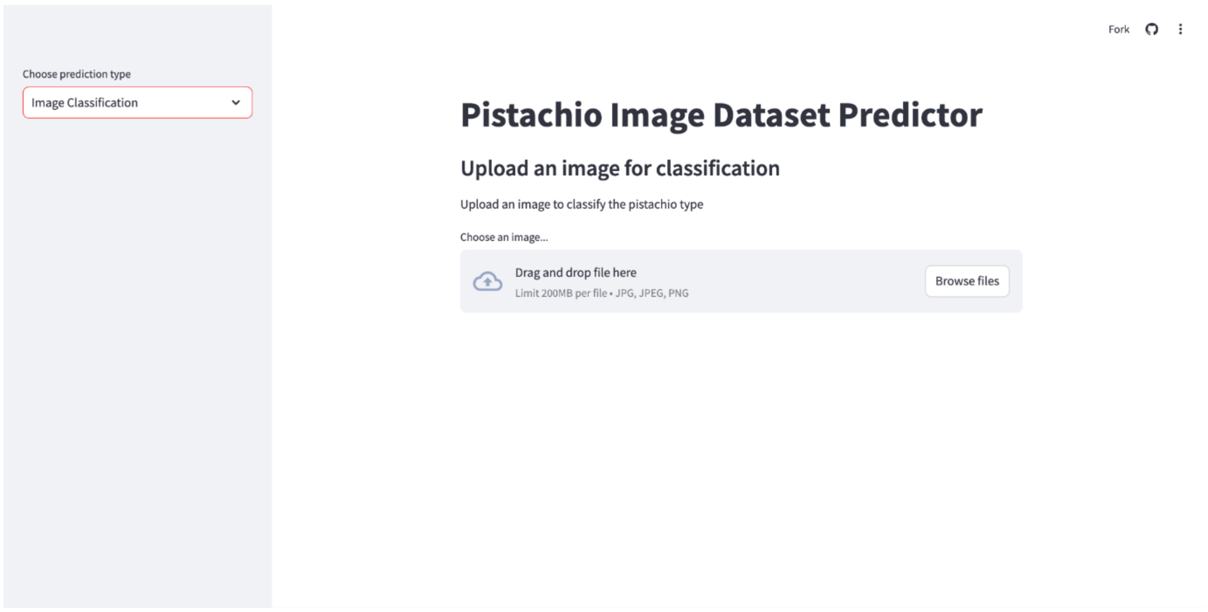
Extent (Standard)
0.10
0.00 1.00

Aspect Ratio (Standard)
3.80
1.00 4.00

Prediction: 0

Fig. 4.22 Numerical input for 28 features

Fig. 4.22 shows the home page for the numerical feature input. Fig. 4.21 shows input page for 28 features numerical data.



Choose prediction type
Image Classification

Pistachio Image Dataset Predictor

Upload an image for classification

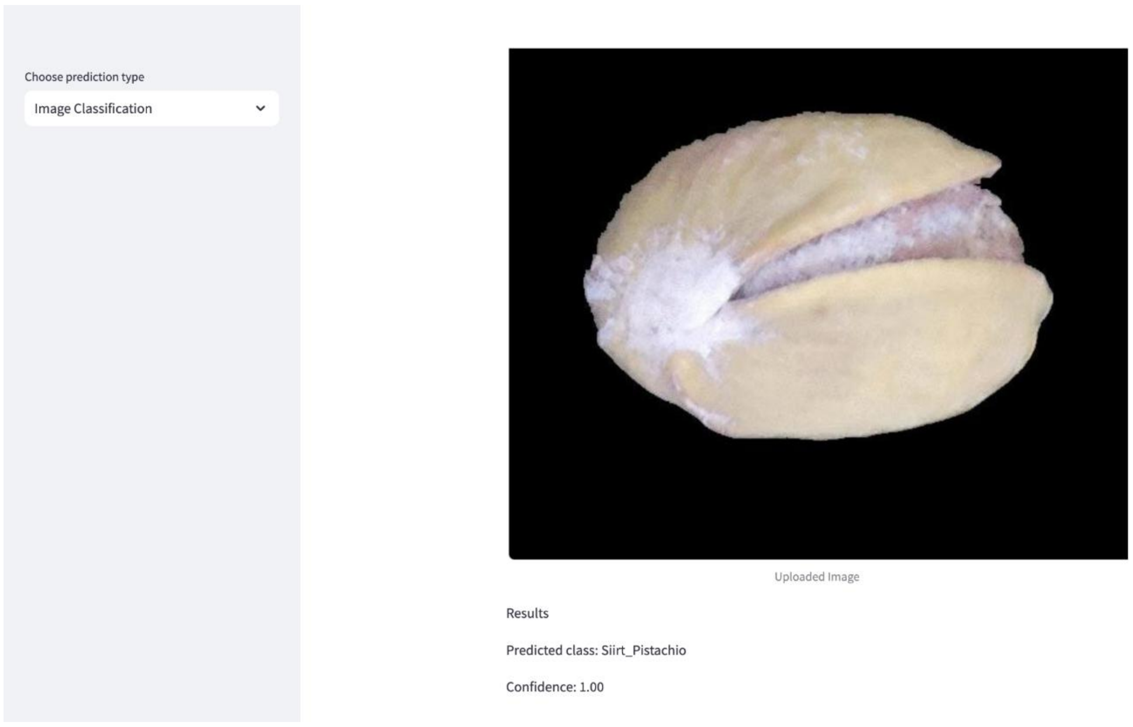
Upload an image to classify the pistachio type

Choose an image...

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Fig. 4.23 Home page for image classification

Fig. 4.23 shows the home page of the image classification. Fig. 4.24 and Fig. 4.25 shows the classification of the pistachios using image classification.

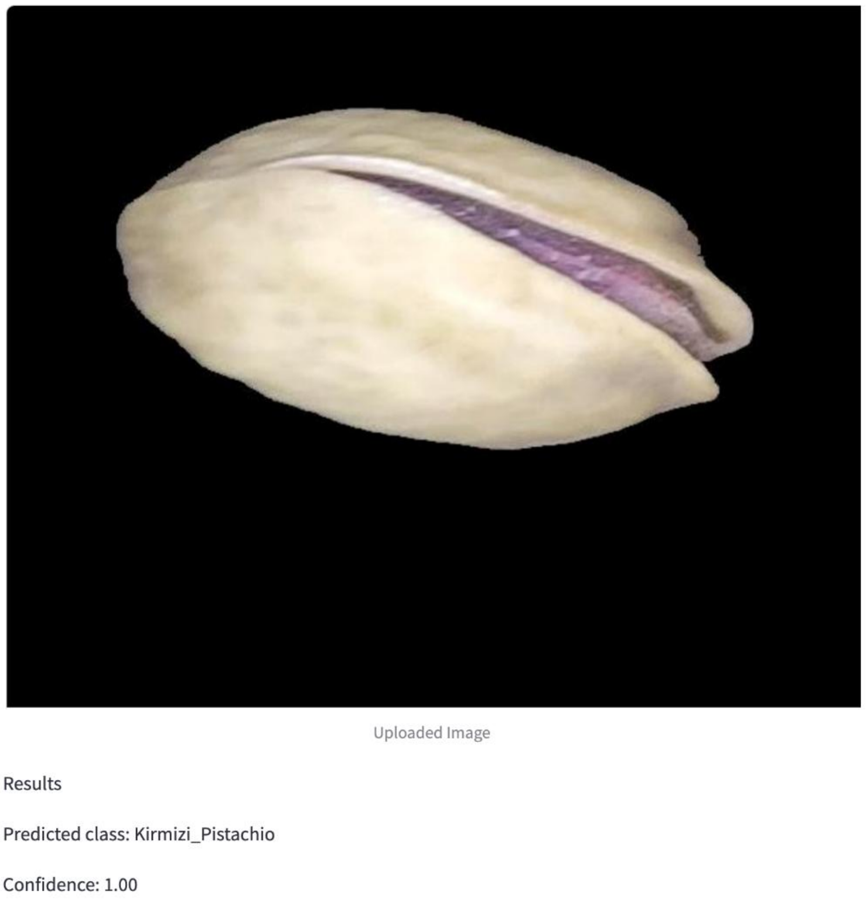


Results

Predicted class: Siirt_Pistachio

Confidence: 1.00

Fig. 4.24 Classification of Siirt Pistachio



Results

Predicted class: Kirmizi_Pistachio

Confidence: 1.00

Fig. 4.25 Classification of Kirmizi Pistachio

CHAPTER 5

TESTING

5.1 Result Table

- Machine Learning (Numerical Data) – 16 Features

Table 5.1 Result Table for Numerical Data – 16 Features

Classifier	Accuracy	Precision	Recall	F1 Score
Logistic Regression	**0.864**	0.865	0.864	0.864
K Nearest Neighbour	**0.843**	0.843	0.843	0.843
Decision Tree	**0.820**	0.820	0.820	0.820
Random Forest	**0.870**	0.871	0.870	0.870
AdaBoost	**0.881**	0.882	0.881	0.881
Gradient Boosting	**0.882**	0.882	0.882	0.882
Extra Trees	**0.853**	0.854	0.853	0.853
Extreme Gradient Boosting	**0.860**	0.861	0.860	0.861
Light Gradient Boosting	**0.864**	0.864	0.864	0.864
CatBoost Classifier	**0.879**	0.880	0.879	0.879
Naïve Bayes	**0.848**	0.852	0.848	0.849
Quadratic Discriminant Analysis	**0.856**	0.870	0.856	0.857
SVM - Linear Kernel	**0.873**	0.873	0.873	0.873

- Machine Learning (Numerical Data) – 28 Features

Table 5.2 Result Table for Numerical Data – 28 Features

Classifier	Accuracy	Precision	Recall	F1 Score
Logistic Regression	**0.902**	0.902	0.902	0.902
K Nearest Neighbour	**0.840**	0.840	0.840	0.840
Decision Tree	**0.829**	0.831	0.829	0.830
Random Forest	**0.909**	0.908	0.909	0.908
AdaBoost	**0.905**	0.906	0.905	0.906
Gradient Boosting	**0.913**	0.914	0.913	0.913
Extra Trees	**0.902**	0.902	0.902	0.902
Extreme Gradient Boosting	**0.926**	0.926	0.926	0.926
Light Gradient Boosting	**0.919**	0.920	0.919	0.920
CatBoost Classifier	**0.936**	0.937	0.936	0.937
Naïve Bayes	**0.874**	0.883	0.874	0.875
Quadratic Discriminant Analysis	**0.905**	0.911	0.905	0.906
SVM - Linear Kernel	**0.915**	0.915	0.915	0.915

- Fast.ai (Image Data)

Table 5.3 Result Table for Fast.ai models

Sr. No.	Model	Accuracy	Precision	Recall	F1 Score	Error Rate
1	Resnet18	**0.983**	0.993	0.971	0.982	0.017
2	Resnet34	**0.995**	0.994	0.997	0.995	0.005
3	Resnet50	**0.953**	0.911	1.000	0.953	0.047
4	Resnet101	**0.943**	0.899	0.990	0.943	0.057
5	Resnet152	**0.963**	0.930	0.997	0.962	0.037
6	VGG11_bn	**0.977**	0.993	0.958	0.975	0.023
7	VGG13_bn	**0.986**	0.984	0.987	0.985	0.014
8	VGG16_bn	**0.991**	0.993	0.987	0.990	0.009
9	VGG19_bn	**0.995**	0.997	0.993	0.995	0.005
10	DenseNet121	**0.991**	0.993	0.987	0.990	0.009
11	DenseNet161	**0.989**	0.990	0.987	0.989	0.011
12	DenseNet169	**0.977**	0.983	0.967	0.975	0.023
13	DenseNet201	**0.990**	0.990	0.990	0.990	0.009
14	AlexNet	**0.988**	0.993	0.980	0.987	0.012
15	SqueezeNet1_0	**0.935**	0.975	0.886	0.928	0.065
16	SqueezeNet1_1	**0.977**	0.983	0.967	0.975	0.023

- Timm (Image Data)

Table 5.4 Result Table for Timm models

Sr. No.	Model	Accuracy	Precision	Recall	F1 Score	Error Rate
1	xception	**0.989**	0.981	0.997	0.989	0.011
2	efficientnet_b0	**0.974**	0.948	1.00	0.973	0.026
3	efficientnet_b1	**0.98**	0.959	1.00	0.979	0.02
4	efficientnet_b2	**0.995**	0.99	1.00	0.995	0.005
5	efficientnet_b3	**0.988**	0.975	1.00	0.987	0.012
6	efficientnet_b4	**0.988**	0.984	0.990	0.987	0.012
7	efficientnet_b5	**0.753**	0.898	0.544	0.677	0.247
8	regnetx_002	**0.983**	0.965	1.00	0.982	0.017
9	regnetx_004	**0.989**	0.978	1.00	0.989	0.011
10	regnetx_006	**0.974**	0.948	1.00	0.973	0.026
11	regnetx_008	**0.992**	0.987	0.997	0.992	0.008
12	regnetx_016	**0.981**	0.962	1.00	0.981	0.019
13	regnetx_032	**0.981**	0.962	1.00	0.981	0.019
14	regnetx_040	**0.989**	0.978	1.00	0.989	0.011

15	regnetx_064	**0.989**	0.978	1.00	0.989	0.011
16	regnetx_080	**0.980**	0.959	1.00	0.979	0.02
17	regnetx_120	**0.964**	0.930SSS	1.00	0.964	0.036
18	regnetx_160	**0.943**	0.892	1.00	0.943	0.057
19	regnetx_320	**0.958**	0.919	1.00	0.958	0.042

5.2 Result Analysis

In the numerical data when trained for 16 features (excluding the colour features) shown in Table 5.1, SVM achieved the highest accuracy of 87.4% among the ML algorithms. On using ensemble techniques, Gradient Boosting outperformed the other algorithms highest accuracy of 88.2% with precision, recall and F1 of 88%. On the other hand, when the colour features were used in training, CatBoost achieved the highest accuracy 93.6% (shown in Table 5.2) outperforming the other algorithms. The increase in the performance shows the significance of colour features in the model training.

In transfer learning from Fast.ai and Timm library on the image data, there was boost in the performance of classification compared to models learning from scratch. In Fast.ai, ResNet34 outperformed with an accuracy of 0.995 and error rate of 0.005. ResNet18 achieved an accuracy of 0.98. ResNet50 had an accuracy of 0.953 and error rate of 0.047. ResNet101 and ResNet152 showed accuracies of 0.943 and 0.963. VGG19_bn had the highest accuracy of 0.995, while VGG16_bn scored 0.991. VGG11_bn and VGG13_bn models achieved accuracies of 0.977 and 0.986. Among DenseNet models, DenseNet121 and DenseNet201 had similar accuracies of 0.991, while DenseNet161 and DenseNet169 had accuracies of 0.989 and 0.977. AlexNet achieved 0.988 accuracy. SqueezeNet1_0 and SqueezeNet1_1 models showed accuracies of 0.935 and 0.977, respectively (shown in Table 5.3). In Timm, Xception achieved accuracy of 0.989, Efficientnet_b2 outperformed its variants with accuracy of 0.995 and a very low error rate of 0.005 and Regnetx_008 achieved accuracy of 0.992 shown in Table 5.4.

- **Highest Accuracy in ResNet Models**: **ResNet34** with 0.995.
- **Highest Accuracy in VGG Models**: **VGG16_bn** with 0.991.
- **Highest Accuracy in DenseNet Models**: **DenseNet201** with 0.988.
- **Highest Accuracy in SqueezeNet Modes**: **SqueezeNet1_1** with 0.977.
- **Highest Accuracy in EfficientNet Models: Efficientnet_b2** with 0.995.
- **Highest Accuracy in RegNetX Models: RegNetx_008** with 0.992.

CHAPTER 6

CONCLUSION & OUTCOMES

6.1 Overall Analysis of Project Viabilities

The project successfully demonstrated the power of combining traditional machine learning approaches with modern transfer learning techniques for pistachio classification. Through training with both numerical features (16 and 28) and image data, we found that the 28-feature dataset consistently outperformed the 16-feature dataset, with CatBoost Classifier achieving the highest accuracy of 93.6% among traditional ML models. In deep learning, multiple pre-trained architectures through FastAI and Timm frameworks obtained very good results, with ResNet34, VGG19_bn, and EfficientNet-B2 all achieving 99.5% accuracy. This performance shows the effectiveness of transfer learning in handling agricultural classification tasks. The successful integration of these models into a user-friendly web application which offers both image-based and feature-based classification options not only advances in the field of automated sorting but also provides a scalable, reliable solution for the pistachio industry, by reducing costs and improving quality control in real-world applications.

6.2 Problems encountered and possible solutions

During these projects various problems were encountered. First and the major one is the dataset, which was small meaning models were not training as efficiently as possible. Also, the dataset while being small was too simple to train, the classes were clearly distinguishable and only few of the samples had similarities which could help the model train properly. Pre-processing was done to ensure proper model learns properly. Second, the models were prone to overfitting because of simple features. Although pre-processing was applied, the dataset was still simple for complex deep learning architectures used in transfer learning. Because of these the models were not able to properly train after certain epochs. Using a larger and more complex dataset could overcome this problem. Hyperparameter tuning was also tried but it didn’t provide useful results. Lastly, in the web application for numerical features, the input type was taken slider, which is not user-friendly, tedious and time consuming. Also, there are some cases in image classification where there were some misclassifications (there is no pistachio in the image, still it predicts with high confidence). Replacing the sliders with the text input field and implementing confidence thresholding where predictions below a certain confidence score would trigger a "Cannot Classify/Invalid Image" message rather than forcing a classification could overcome these issues.

6.3 Project Outcomes

First, we created a, dual-approach classification technique that accommodates both feature-based and image-based classification methods, compatible for different user needs and data availabilities. Second, we developed an accessible, web application that transforms the models into a practical tool usable without technical expertise. Further our model architectures, preprocessing techniques, and implementation strategies serves as a valuable resource for researchers and developers working on related projects. Lastly, we showed the advantages of using transfer learning, potentially reducing the need for training models from scratch in similar projects.

6.4 Future Scope

The future scope of this project will primarily focus on optimizing the fine-tuning process for smaller datasets through advanced transfer learning techniques and implementing hybrid approaches that combine both feature-based and image-based classification methods. Along with this, refinement of the web-application by implementing a more user-friendly interface by replacing the current slider-based inputs with intuitive text fields and a better classifying web-based application by applying automatic "Cannot Classify" response for cases where prediction confidence falls below acceptable thresholds. To address the dataset limitations and overfitting issues, our focus will be on combining multiple pistachio datasets and possibly combining samples from various conditions, and varieties. This expansion with refined model architectures, will enhance the system's generalization abilities while maintaining high classification accuracy.

References

[1] Y. Özçevik, “A Real-Time Nut-Type Classifier Application Using Transfer Learning,” *Appl. Sci.*, vol. 13, no. 21, 2023, doi: 10.3390/app132111644.

[2] A. S. Sabah and S. S. Abu-naser, “Pistachio Variety Classification using Convolutional Neural Networks,” vol. 8, no. 4, pp. 113–119, 2024.

[3] E. AVUÇLU, “Classification of Pistachio Images Using VGG16 and VGG19 Deep Learning Models,” *Int. Sci. Vocat. Stud. J.*, vol. 7, no. 2, pp. 79–86, 2023, doi: 10.47897/bilmes.1328313.

[4] I. A. Özkan, M. Köklü, and R. Saraçoğlu, “Classification of pistachio species using improved k-NN classifier,” *Prog. Nutr.*, vol. 23, no. 2, pp. 1–9, 2021, doi: 10.23751/pn.v23i2.9686.

[5] F. Patel, S. Mewada, S. Degadwala, and D. Vyas, “Recognition of Pistachio Species with Transfer Learning Models,” *Int. Conf. Self Sustain. Artif. Intell. Syst. ICSSAS 2023 - Proc.*, no. Icssas, pp. 250–255, 2023, doi: 10.1109/ICSSAS57918.2023.10331907.

[6] Q. Sun, “Food Image Classification and Food Safety Detection based on Pretrained Convolutional Neural Network,” pp. 01–06, 2024, doi: 10.1109/icdcece60827.2024.10548096.

[7] A. Kaur, V. Kukreja, D. Upadhyay, M. Aeri, and R. Sharma, “An Effective Pistachio Classification by Ensembling Fine-tuned ResNet20 and DenseNet Models,” *2024 IEEE Int. Conf. Interdiscip. Approaches Technol. Manag. Soc. Innov. IATMSI 2024*, vol. 2, pp. 1–6, 2024, doi: 10.1109/IATMSI60426.2024.10502708.

[8] R. Pillai, N. Sharma, and R. Gupta, “Classification of Pista Species using Fine-Tuned EfficientNetB3 Transfer Learning Model,” *2023 IEEE Int. Conf. Res. Methodol. Knowl. Manag. Artif. Intell. Telecommun. Eng. RMKMATE 2023*, pp. 1–5, 2023, doi: 10.1109/RMKMATE59243.2023.10369347.

[9] E. Avuçlu, “Classification Of Pistachio Images With The ResNet Deep Learning Model,” *Selcuk J. Agric. Food Sci.*, vol. 37, pp. 291–300, 2023, doi: 10.15316/sjafs.2023.029.

[10] Ç. Gökalp and Ç. M. Mazhar, “An improved pistachio detection approach using YOLO-v8 Deep Learning Models,” *BIO Web Conf.*, vol. 85, pp. 0–4, 2024, doi: 10.1051/bioconf/20248501013.

[11] L. Lisda, K. Kusrini, and D. Ariatmanto, “Classification of Pistachio Nut Using Convolutional Neural Network,” *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 8, no. 1, pp. 71–77, 2023, doi: 10.25139/inform.v8i1.5685.

[12] D. Singh *et al.*, “Classification and Analysis of Pistachio Species with Pre-Trained Deep Learning Models,” *Electron.*, vol. 11, no. 7, pp. 1–14, 2022, doi: 10.3390/electronics11070981.

[13] https://www.kaggle.com/datasets/muratkokludataset/pistachio-image-dataset

[14] https://pistachio-classification.streamlit.app/