Skin Lesion Classification using ML Models

FCV Mini-project

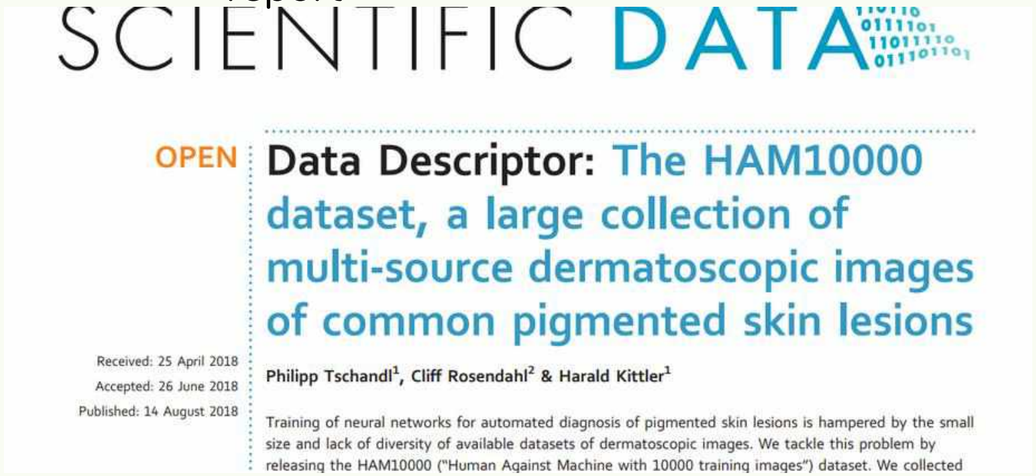
Aditya Gosavi - 54 - 220962356
Kavish Shah - 66 - 220962398
Shaurya Gupta - 51 - 220962332
Kabir Bhalodi - 71 - 220962424

Introduction

- Skin cancer is common and early detection is crucial for effective treatment.
- Traditional methods are time-consuming and require expert knowledge.
- Machine learning (ML) offers faster, scalable, and consistent skin lesion classification.
- By analyzing key image features like color, texture, and shape, ML models can identify patterns in large datasets like HAM10000, a comprehensive collection of dermatoscopic images.

Literary Review

Refer [1] and [2] in the report



SCIENTIFIC DATA

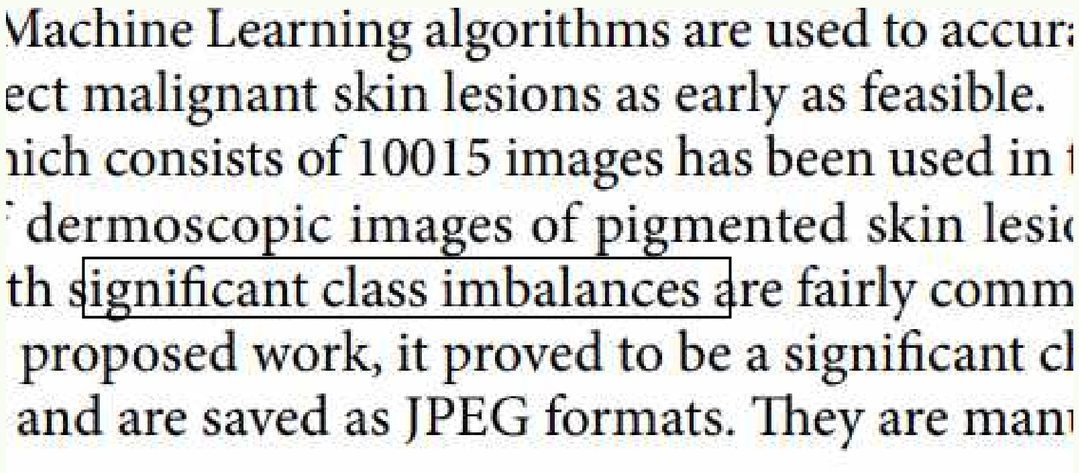
OPEN

Data Descriptor: The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions

Received: 25 April 2018
Accepted: 26 June 2018
Published: 14 August 2018

Philipp Tschandl[1], Cliff Rosendahl[2] & Harald Kittler[1]

Training of neural networks for automated diagnosis of pigmented skin lesions is hampered by the small size and lack of diversity of available datasets of dermatoscopic images. We tackle this problem by releasing the HAM10000 ("Human Against Machine with 10000 training images") dataset. We collected



Machine Learning algorithms are used to accur
ect malignant skin lesions as early as feasible.
hich consists of 10015 images has been used in
dermoscopic images of pigmented skin lesi
th significant class imbalances are fairly comm
proposed work, it proved to be a significant cl
and are saved as JPEG formats. They are man

Dataset Selection

Datasets like HAM10000 provide a large, labeled collection of dermoscopic images, which are crucial for developing and testing machine learning models. This dataset is widely used due to its diversity and comprehensive annotations.

Common Challenges

Key challenges include class imbalance in datasets, the need for robust feature extraction methods, and the high variability in lesion appearances across different skin types. Data augmentation is often used to address class imbalance.

Literary Review

Refer [1] and [2] in the report

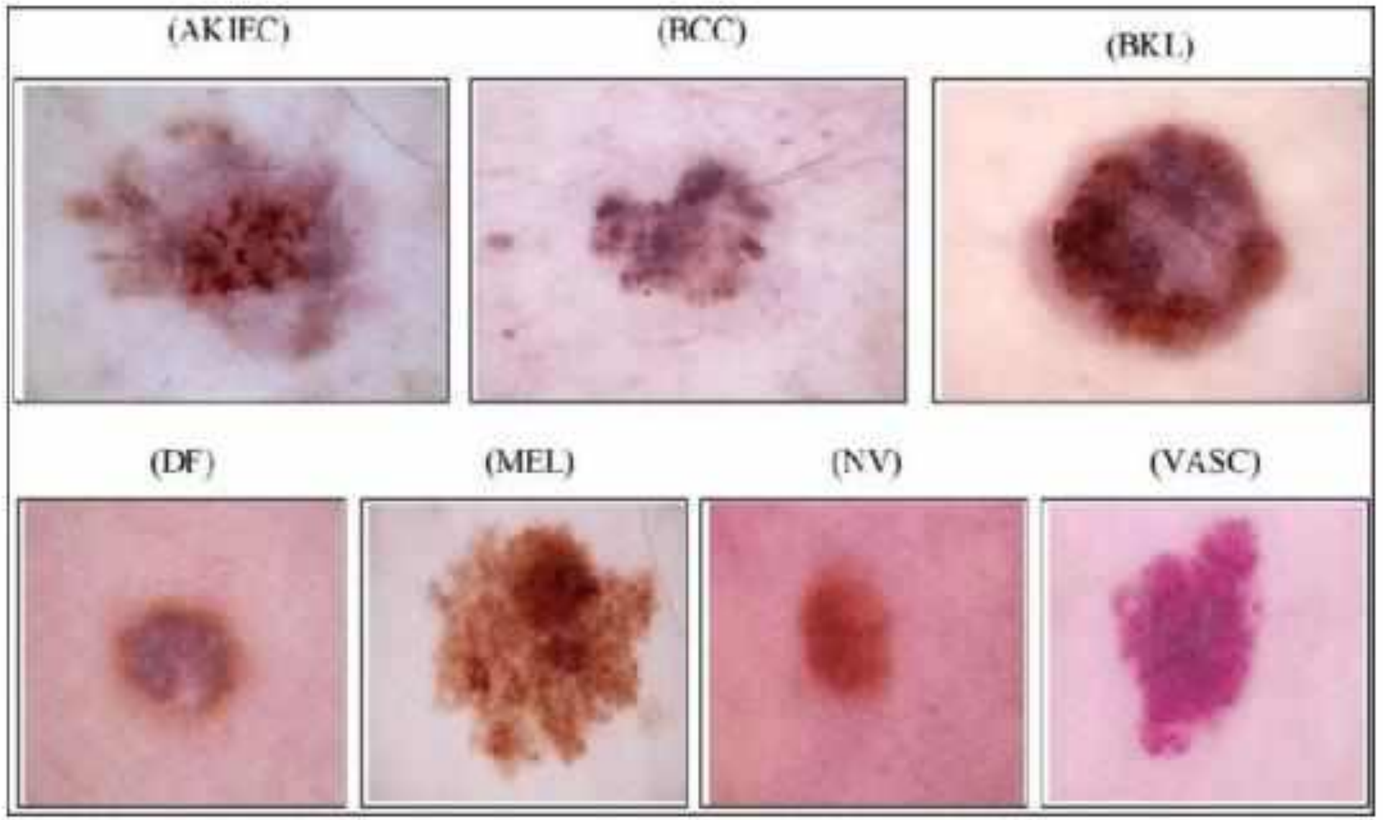


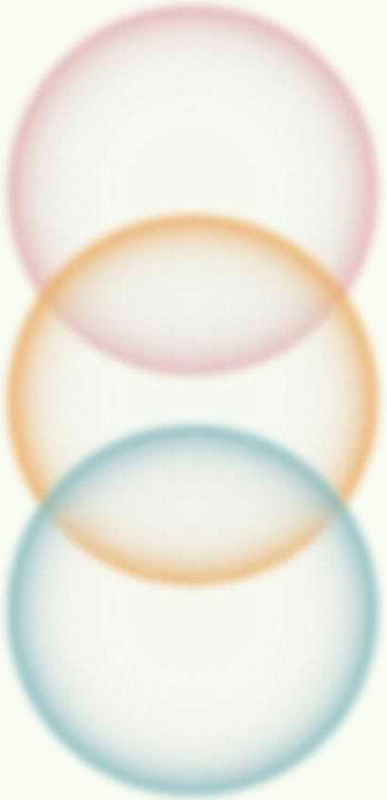
Figure 1. Sample images for the seven skin lesion classes from HAM10000 dataset.

Methodology

A. Data Preprocessing and Augmentation

B. Feature Extraction

C. Model Training and Classification



A. Data Preprocessing and Augmentation

- **Data Loading**: Images loaded from the directory with metadata extracted from the CSV file. Each image is cropped around the lesion.
- **Data Augmentation**: Horizontal flipping used to address class imbalance, doubling the dataset size and enhancing classifier generalization.

B. Feature Extraction

Color features

- The **color histogram** of each image is computed to capture the distribution of pixel intensities, which is important for identifying different lesion types based on color differences.

Texture features

- : **Haralick texture features** are extracted using the mahotas library, which quantifies texture patterns within an image and is particularly useful for distinguishing lesion types based on surface texture.

Shape features

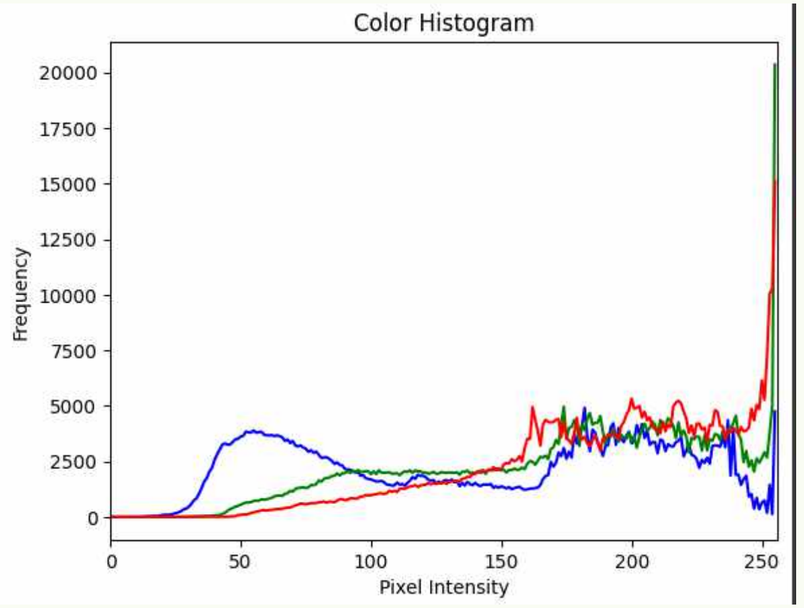
- **Hu Moments** are used to capture the shape of the lesions, since they are invariant to image transformations such as scaling, rotation, and translation, making them robust features for shape classification.

B. Feature Extraction

Color features

- The **color histogram** of each image is computed to capture the distribution of pixel intensities, which is important for identifying different lesion types based on color differences.

```python
def extract_color_histogram(image, bins=(8, 8, 8)):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([hsv], [0, 1, 2], None, bins, [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()
    return hist
```

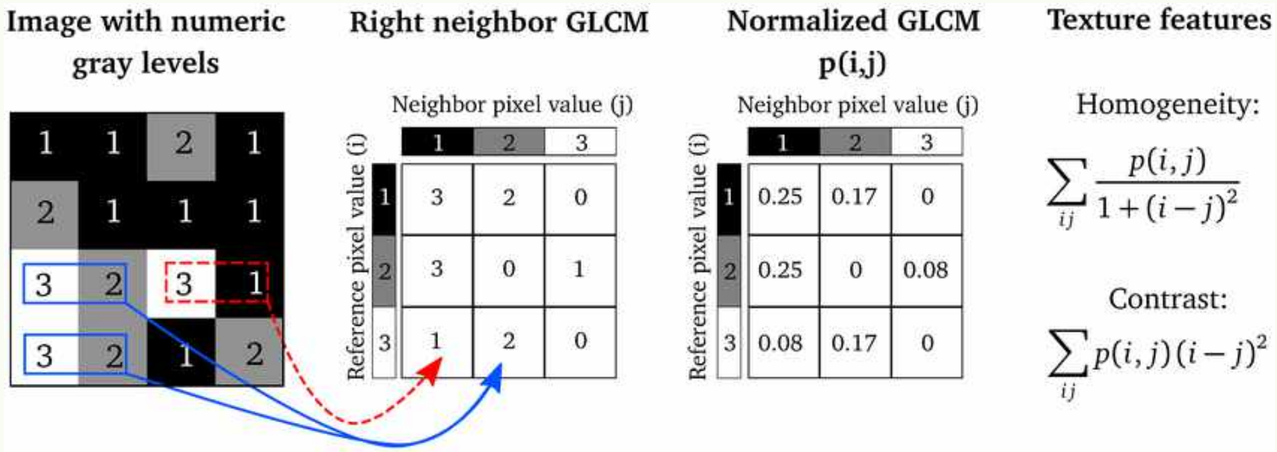


B. Feature Extraction

Texture features

- **Haralick texture features** are extracted using the mahotas library, which quantifies texture patterns within an image and is particularly useful for distinguishing lesion types based on surface texture.

```python
def extract_haralick_texture(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    texture = mahotas.features.texture.haralick(gray)
    return texture.mean(axis=0)
```

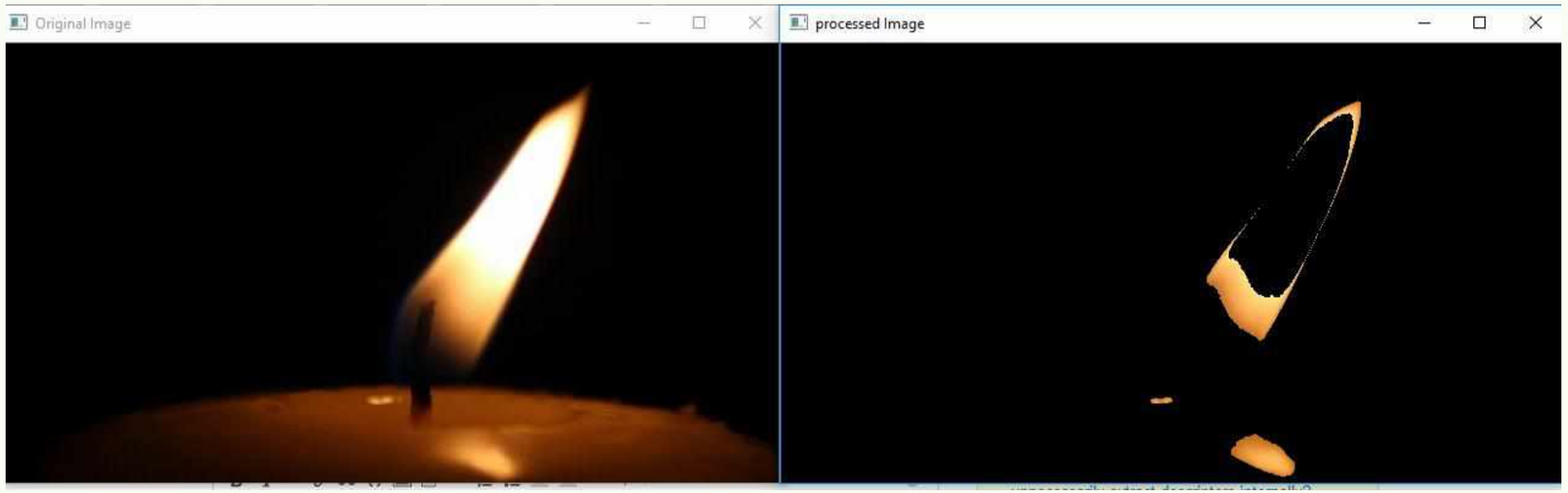


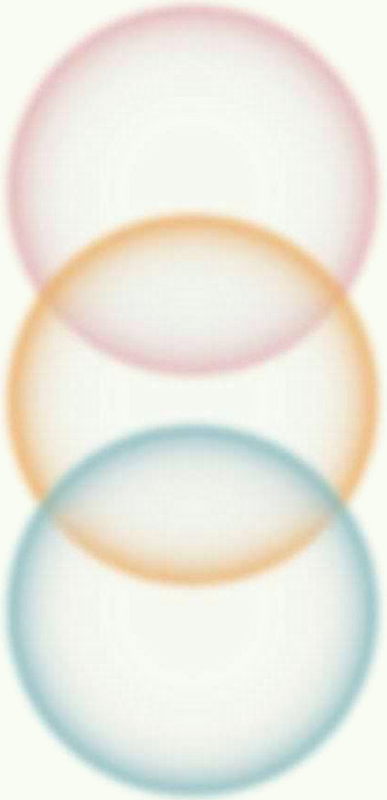
B. Feature Extraction

Shape features

- **Hu Moments** are used to capture the shape of the lesions, since they are invariant to image transformations such as scaling, rotation, and translation, making them robust features for shape classification.

```python
def extract_hu_moments(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    moments = cv2.moments(binary)
    hu_moments = cv2.HuMoments(moments).flatten()
    return hu_moments
```





C. Model Training and Classification

- **Decision Tree Classifier:** Trains on extracted features, capable of learning complex decision boundaries and providing interpretability.
- **Random Forest Classifier:** Uses an ensemble of decision trees for better accuracy and reduced overfitting by aggregating predictions.
- **K-Nearest Neighbors (KNN) Classifier:** A non-parametric classifier that determines the class based on majority vote from k nearest neighbors, trained using feature vectors.

Evaluation Metrics

- **Accuracy:** The accuracy of each model is computed, and a classification report is generated to evaluate precision, recall, and F1-score for each class.

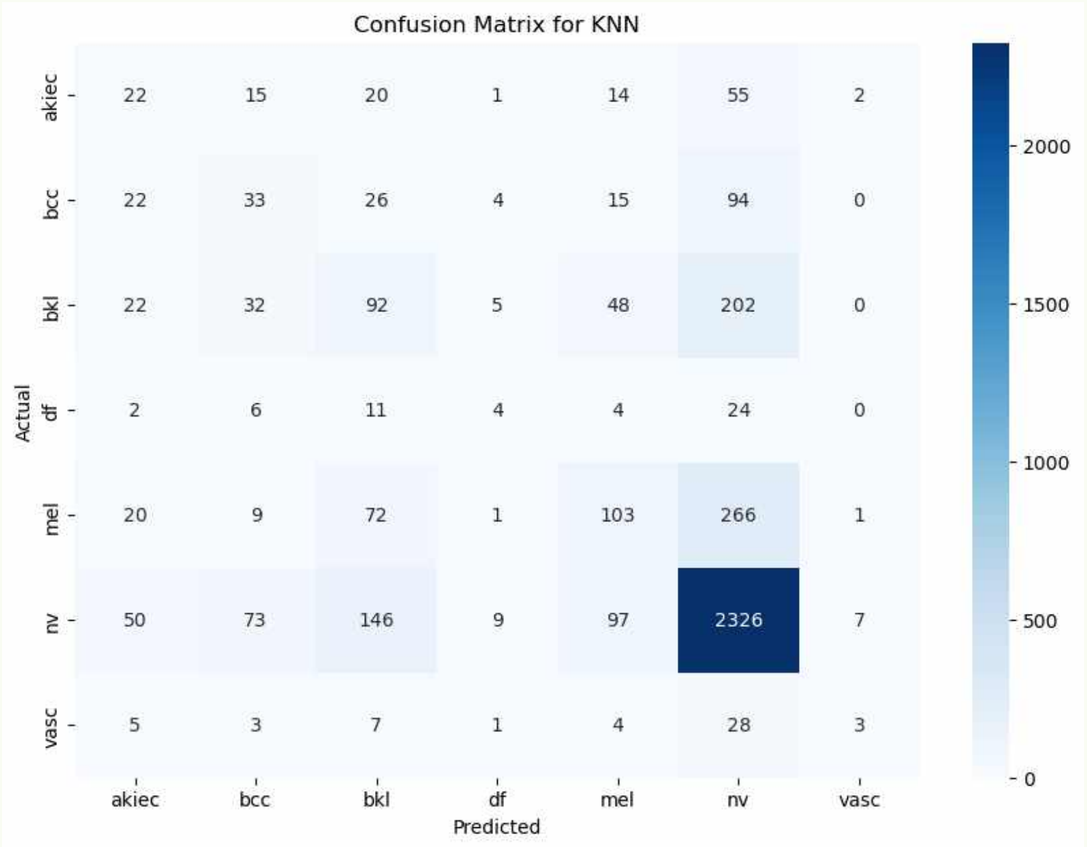
- **Confusion Matrix:** A confusion matrix is plotted to visualize the model's performance in classifying different lesion types. It shows the true positive, false positive, true negative, and false negative counts for each class, helping to identify misclassifications.

- **Sample Visualization:** A set of random samples from the test set are displayed alongside their true and predicted labels, allowing visual inspection of the model’s performance on individual cases.

Results - KNN

Accuracy: 64.48%

- **Precision:** KNN struggled with precision
- **Recall:** Low recall for most classes, particularly for rare categories
- **F1-Score:** Very low F1-scores, especially for rare classes
- **Strengths:** Simple model, but struggles with class imbalances and high-dimensional data.
- **Weaknesses:** Poor overall performance with a significant drop in accuracy and recall



Results - KNN

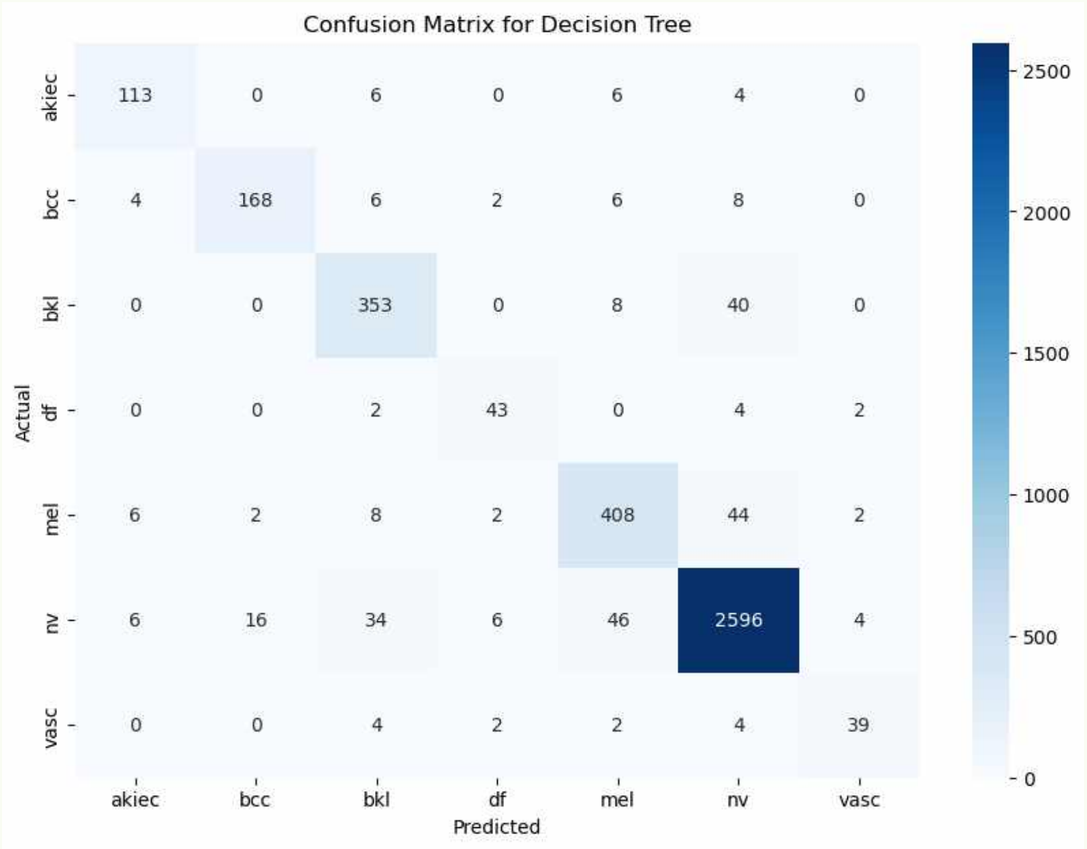
Accuracy: 64.48%

- **Precision:** KNN struggled with precision
- **Recall:** Low recall for most classes, particularly for rare categories
- **F1-Score:** Very low F1-scores, especially for rare classes
- **Strengths:** Simple model, but struggles with class imbalances and high-dimensional data.
- **Weaknesses:** Poor overall performance with a significant drop in accuracy and recall

Results - Decision Trees

Accuracy: 92.86%

- **Precision**: High precision for most classes.
- **Recall**: High recall for [nv], moderate recall for other common classes
- **F1-Score**: Balanced F1-scores across the categories
- **Strengths**: Simple, interpretable model with good accuracy and performance
- **Weaknesses**: Struggles slightly with rare classes due to class imbalance.



Results - Decision Trees

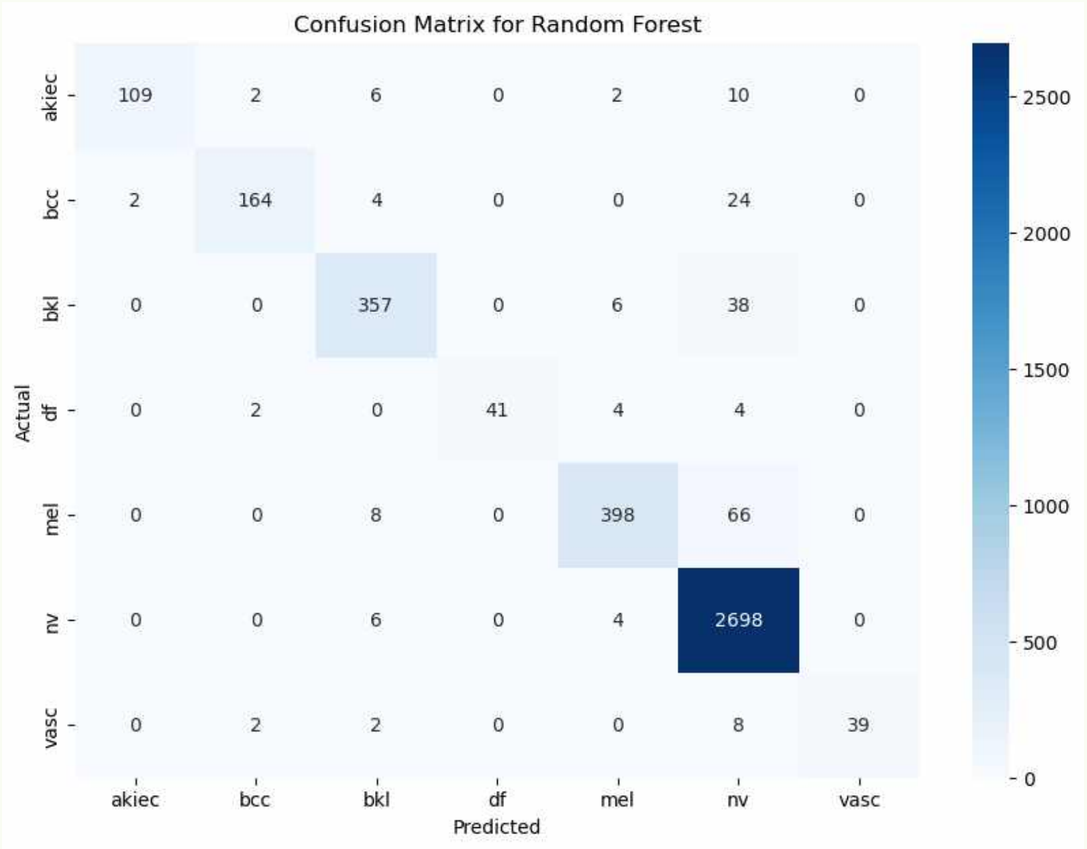
Accuracy: 92.86%

- **Precision**: High precision for most classes.
- **Recall**: High recall for [nv], moderate recall for other common classes
- **F1-Score**: Balanced F1-scores across the categories
- **Strengths**: Simple, interpretable model with good accuracy and performance
- **Weaknesses**: Struggles slightly with rare classes due to class imbalance.

Results - Random Forest

Accuracy: 95.01%

- **Precision**: Very high precision across most classes
- **Recall**: Excellent recall for [nv] (100%) but slightly lower for rare classes
- **F1-Score**: High F1-scores overall, showing a good balance
- **Strengths**: Robust model with excellent handling of class imbalances
- **Weaknesses**: More complex and less interpretable than Decision Trees.

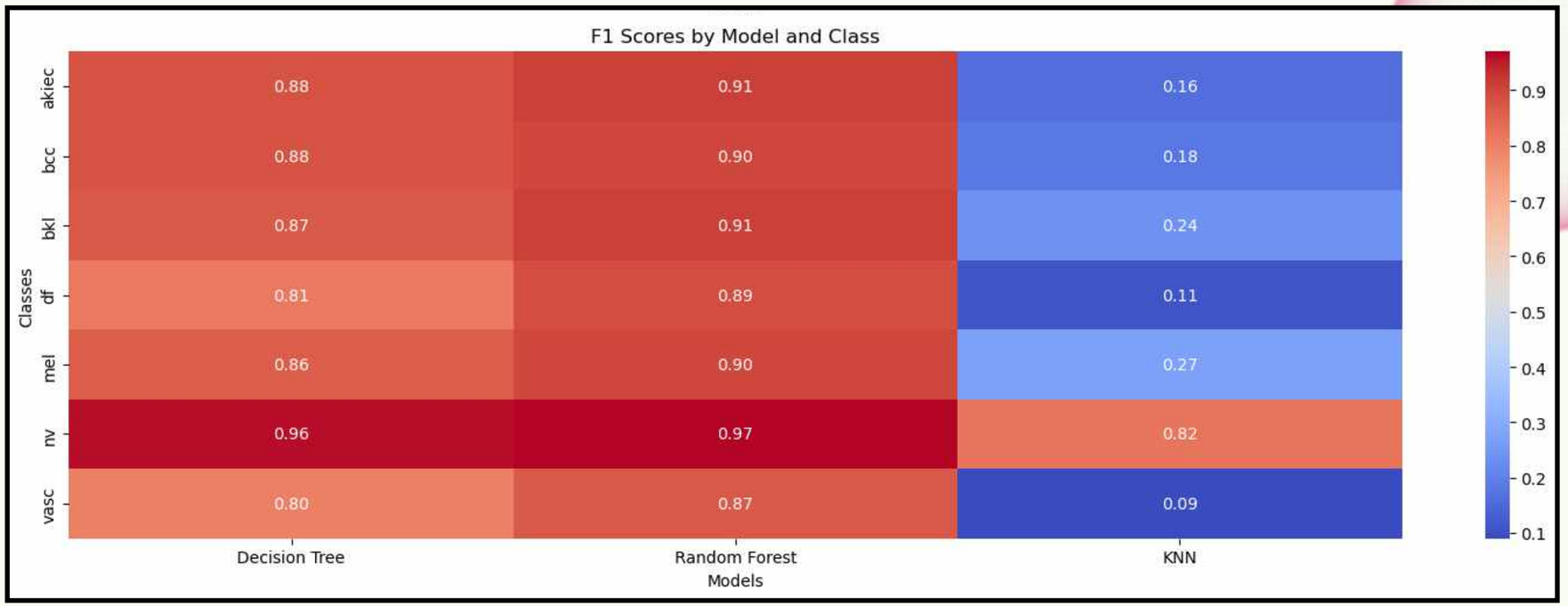


Results - Random Forest

Accuracy: 95.01%

- **Precision**: Very high precision across most classes
- **Recall**: Excellent recall for [nv] (100%) but slightly lower for rare classes
- **F1-Score**: High F1-scores overall, showing a good balance
- **Strengths**: Robust model with excellent handling of class imbalances
- **Weaknesses**: More complex and less interpretable than Decision Trees.

Analysis



Above are the F1 Scores plotted for each Model and Class to see where each model goes wrong. We can clearly see that KNN is particularly weak in predicting classes other than [nv] correctly, whereas Decision Trees and Random Forest are much stronger.

Conclusion

- **Random Forest:** Best performance with 95.01% accuracy, handling both common and rare classes well. Highest precision, recall, and F1-scores across all classes.
- **Decision Tree**: Strong performance with 92.86% accuracy, but less effective for rare classes compared to Random Forest.
- **K-Nearest Neighbors (KNN):** Poor performance with 64.48% accuracy, struggling with precision and recall, especially for rare classes. Least suitable for this imbalanced dataset.
- **Random Forest is the best ML model** for skin lesion classification, followed by Decision Tree. KNN is not recommended for this task due to poor handling of imbalanced data.

Thank you!