# Find The Break - A Computer Vision Algorithm to Detect Fractures on X-ray Images

*Authors: Thomas Dolan, Charlie Glass, Philippe Gagnon, Meir T. Marmor*

*Instructor: Senthil Periaswamy*

**ABSTRACT**

**Objective:** The incidence of bone fractures is increasing globally, even in affluent nations. The accurate diagnosis of bone fractures is vital for effective treatment. However, manual inspection of X-ray images has limitations, highlighted by instances where radiologists, due to fatigue, have overlooked fractures in images[5]. The “Find The Break” project aims to combine computer vision and machine learning techniques into an algorithm that can accurately detect fractures on plain X-ray films.

**Methods:** A dataset from Kaggle, containing 9363 X-ray images evenly divided and labeled as fracture and no fracture was used to train the models. Feature extraction was achieved by a combination of image preprocessing techniques including resizing, noise removal, and contrast enhancement, followed by advanced feature extraction methods such as Gray-Level Co-occurrence Matrix (GLCM), Local Binary Pattern (LBP), and convolutional neural networks via ResNet-50. The algorithm underwent training with an 80:20 split of training to testing data, applying models including Linear Regression, Non-linear SVM, and various CNN configurations.

**Results:** The algorithm demonstrated promising outcomes, with the Non-linear SVM achieving the highest Area Under the Curve (AUC) of 0.97. The most generalizable model, combining ResNet and LBP features, attained a validation accuracy of 79%. Visualization tools like TSNE and PCA provided insights into feature separability and dimensional reduction, aiding in model optimization.

**Conclusion**: "Find The Break" shows potential in using computer vision and machine learning to improve the diagnostic accuracy of bone fractures from X-ray images. Further improvements are expected by expanding the dataset and refining feature extraction and model training processes, enhancing the algorithm's applicability in clinical settings.

**Keywords:** computer vision, machine learning, X-ray image analysis, fracture detection, CNN, SVM, diagnostic accuracy

## INTRODUCTION

The human body, comprising 206 bones of varying sizes and shapes, is subject to fractures that can occur in anyone at any time, making early detection and treatment crucial. The incidence of bone fractures is increasing globally, even in affluent nations[1]. Since their discovery in 1895, X-rays have remained a primary tool for diagnosing bone fractures due to their accessibility, cost-effectiveness, and rapid results[2]. The introduction of digital imaging, facilitated by the Digital Imaging and Communications in Medicine (DICOM) standard, has led to advancements in medical imaging by integrating digital X-ray machines and computerized image processing[3]. Adopting machine learning in medical imaging analysis has emerged as a significant advancement, with machine learning increasing in identifying skeletal abnormalities[4].

The accurate diagnosis of bone fractures is vital for effective treatment. However, manual inspection of X-ray images has limitations, highlighted by instances where radiologists, due to fatigue, have overlooked fractures in images[5]. Automated diagnostic tools have become increasingly desirable to minimize human error, with computer vision systems now capable of screening X-ray images for anomalies and alerting physicians. This project aims to combine computer vision and machine learning techniques into an algorithm that can accurately detect fractures on plain X-ray films. With millions of fractures reported annually worldwide, this kind of computer-aided diagnostic system may lead to improved patient outcomes.

**METHODS**

Our process follows the work of state-of-the-art systems in bone fracture detection with four main steps in creating our fracture detection system: preprocessing, edge detection, feature extraction, and classification [6] .

**Source Data and Preprocessing**

We sourced our X-ray image data set from Kaggle user Vuppala Adithya Sairam [8]. This archived set of images contained two folders: training (8863 images) and validation (500) images. We reserved the validation folder for the test set withheld from training and randomly selected a balanced amount of X-rays from the training to make an 80:20 split of training and testing data. The train and validation folders each contained a balanced set of fractured and normal X-ray images in JPEG format. This dataset has used image rotation to augment the data, with each original X-ray image with a fracture having been rotated 39 times and each normal X-ray rotated 59 times, making for a total of 40 or 60 images for each X-ray dependent on if they were fractured. Rotated images are of size 224 x 224 pixels and to maintain consistency in sizing across all images, we resized the original X-ray image to 224 x 224 as well.

After resizing, further steps were taken in preprocessing, starting with loading images in grayscale and beginning noise removal. Noise removal in its many forms is standard in computer vision applications. X-ray images, in particular, are prone to salt and pepper noise, which are pixels with the value 255 (white) or 0 (black) that otherwise should not be [6]. To address this noise, we implemented a **Gaussian filter**, tuning the filter's parameters until 0 and 255 pixels were eliminated and no further, as any more blurring than necessary would impact edge detection.

With noise filtered, the last step in preprocessing is **contrast enhancement**, which may improve the results of our next step, **edge detection**. This was run using Canny Edges implementation. The goal of this is to reduce the amount of data that needs to be incorporated into features.

**Feature Extraction**

**Feature 1: GLCM**

Gray-level Co-occurrence Matrix (GLCM) is a texture descriptor. It measures how often pairs of pixels with specific values and spatial relationships occur within an image. The features we extracted describe several types of relationships among pixels, such as contrast, correlation, energy, and homogeneity. The output feature is an array of 140 values. One of the known weaknesses of GLCM is that it can be computationally intense. However, for the dataset and implementation at hand this did not become a binding constraint.

**Feature 2: LBP**

Local Binary Pattern (LBP) is also a texture descriptor. It is often used in computer vision tasks including image classification. Within the field of medical imaging, it lends itself well to black and white images such as x-rays due to its focus on pixel intensity and its representation in histogram (data array) format. Several variants of LBP are cited across

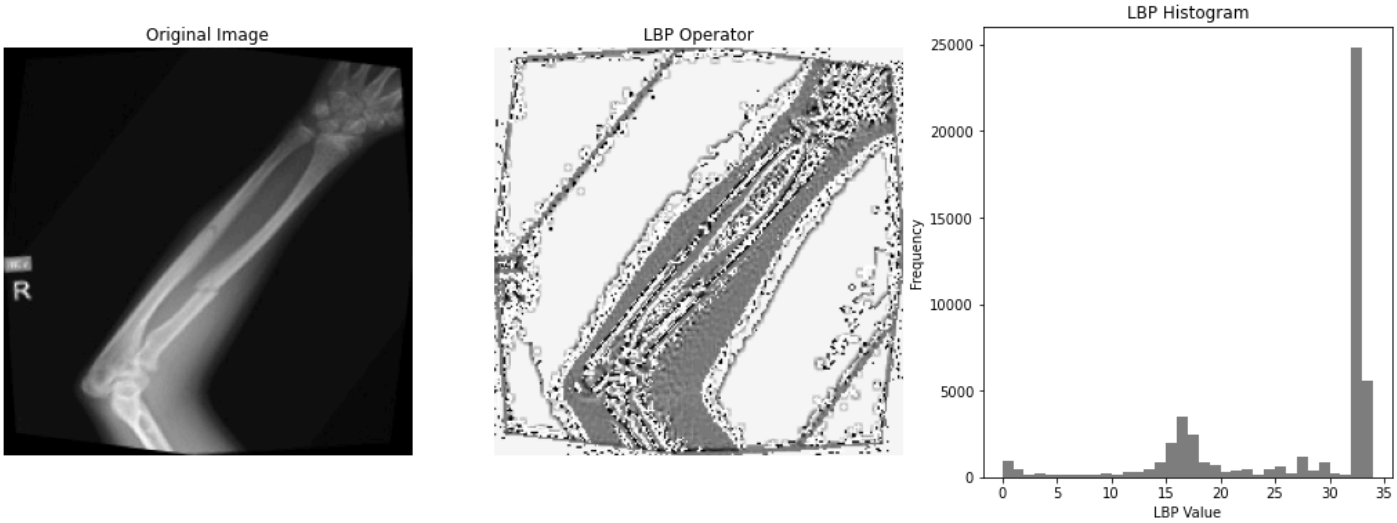
medical literature and other fields [6, 7]. As a first iteration within the context of the course, we prefer using the basic formulation of LBP as implemented in skimage.feature module.

LBP compares each pixel with its neighbor pixels. It then assigns a binary code to the center pixel depending on whether surrounding pixels are greater/smaller than the center pixel's value. Using itemfreq from scipy.stats, the number of occurrences of each binary pattern is transformed into a histogram. This provides both visual intuition of the patterns contained within the image and a feature ready for use within statistical models.

The key parameters to Local Binary Patterns are:

- Radius (size of the circular region of neighboring pixels considered)
- Number of points (number of equidistant points sampled around the circumference of the circle)

Below is an example of an image processed with LBP, along with its corresponding histogram:




The LBP feature produced within this project is an array (histogram) of 34 values, to which standard scaling is applied as a post-processing step.
One of the strengths of LBP is that it computes quickly and uses relative values so it is less sensitive to luminosity. On the other hand, as it focuses on local texture information, larger patterns in the image may be missed. It is also sensitive to noise, hence the relevance of effective smoothing applied during pre-processing steps.

**Feature 3: ResNet**

ResNet-50 is a pre-trained CNN with 50 convolutional layers, one max-pooling layer, and one average-pooling layer. Inputting 224 by 224 RGB images into ResNet-50 returns a 7 by 7 by 2048 numpy array for each image. We used these pre-trained outputs as features for our fractured and non-fractured images to bolster our existing features with inputs from a proven image CNN model trained on much more data.

**Baseline Classifier**

A logistic regression model was employed, using the 'liblinear' solver. The model was trained on the training dataset to classify images based on extracted features.

*Model Evaluation:*

The logistic regression model was evaluated on the testing dataset. Key performance metrics included:

- Accuracy: Calculated as the percentage of correct predictions.
- Confusion Matrix: Visualized to show the counts of true positives, true negatives, false positives, and false negatives.
- Classification Report: Provided a breakdown of precision, recall, and F1-score.
- ROC Curve: Plotted to assess the model's ability to discriminate between classes, with the Area Under Curve (AUC) serving as a summary metric.
- Feature Importance: Assessed by examining the coefficients of the logistic regression model, identifying the most influential features.

*Feature Importance Analysis:*

The coefficients from the logistic regression model were plotted to visualize and interpret the importance of each feature. The most influential feature was identified, and its impact on the model's performance was further analyzed by retraining the model using only this feature.

*Efficiency and Bias Calculation:*

Custom functions were implemented to compute the efficiency and bias of the predictions. Efficiency was defined as the ratio of true positives to actual positives, and bias was calculated as the difference in mean predicted values versus actual values.

*Advanced Analysis:*

The best-performing feature was isolated, and a new logistic regression model was trained. This model's performance was then evaluated and compared against the full-feature model to assess the impact of feature selection on classification accuracy.

**Comparison of Classifiers**

We evaluated five classification methods, each designed to address binary classification problems, implemented through Python's Scikit-Learn library. The classification methods included:

1. Logistic Regression: Employed with 'liblinear' solver and a fixed random state for reproducibility.
2. Linear SVM (Support Vector Machine): Integrated within a pipeline including StandardScaler for feature scaling, and configured with a linear kernel and probability estimation enabled.
3. Linear Non-Separable SVM: Similar to Linear SVM but with a regularization parameter (C) set to 0.1 to allow a softer margin of classification.
4. Nonlinear SVM: Utilized with an RBF (radial basis function) kernel within a scaling pipeline, enabling probability estimates.

5. Linear Discriminant Analysis (LDA): Applied without additional parameters to provide a baseline for linear decision surfaces.

*Training and Evaluation:*

Each model was trained using a designated training set (X_train, y_train) and subsequently evaluated on a test set (X_test, y_test). Model performance was assessed based on the same metrics as the baseline model.

*Metric Comparison and Visualization:*

Key metrics including training efficiency, bias, and timing were compared across models using bar charts, providing a clear visual representation of each model's strengths and weaknesses.

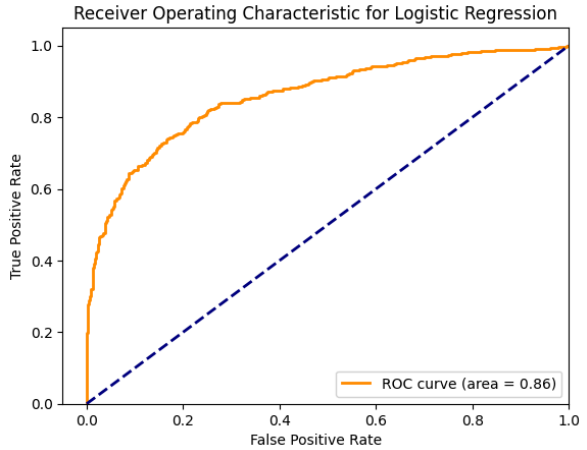
**Software and Libraries**

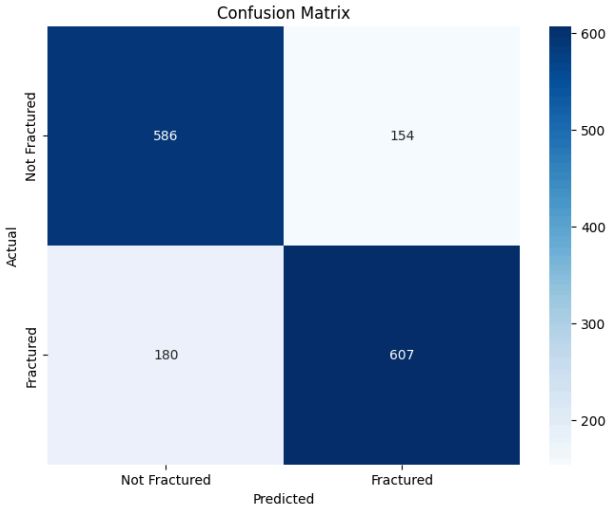
The analysis was conducted using Python (version 3.10.12), with libraries including NumPy, Pandas, Matplotlib, Seaborn, Scikit-image, and Scikit-learn, all of which are integral to data processing, machine learning, and visualization in Python.

**RESULTS**

**Classification - Base model**

The training data was used to train a linear logistic regression model. Model performance was considered very good.
*Full Feature Set - Linear Regression Classifier*



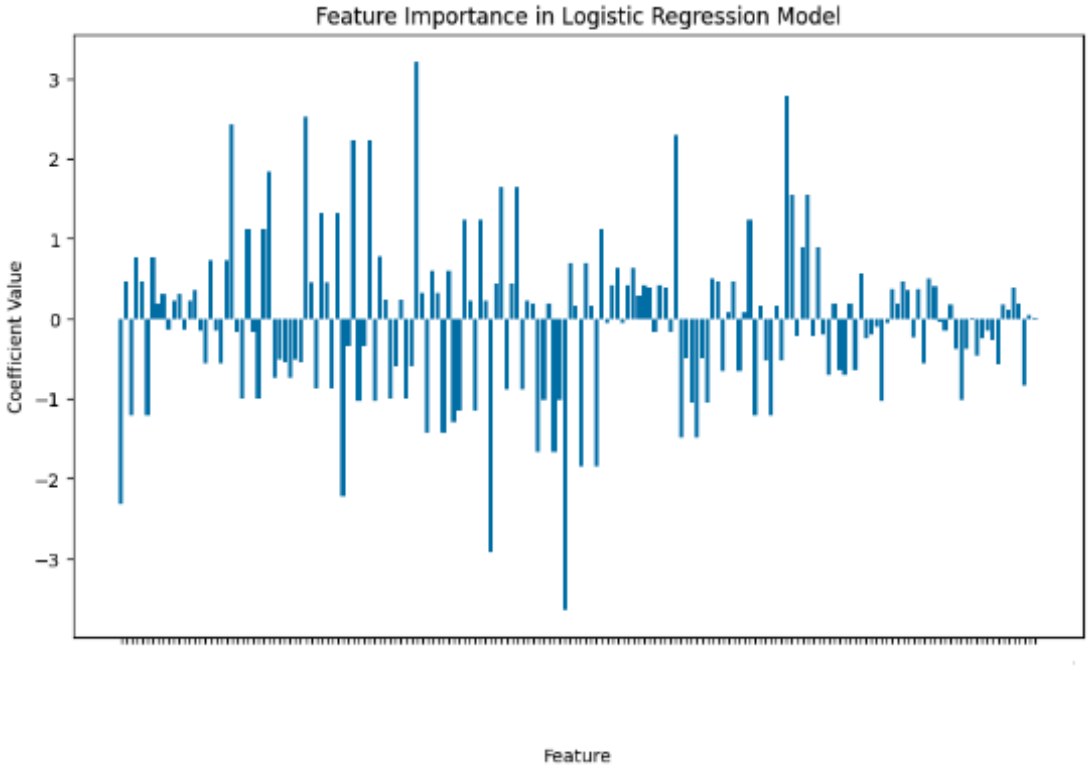




Classification Report:

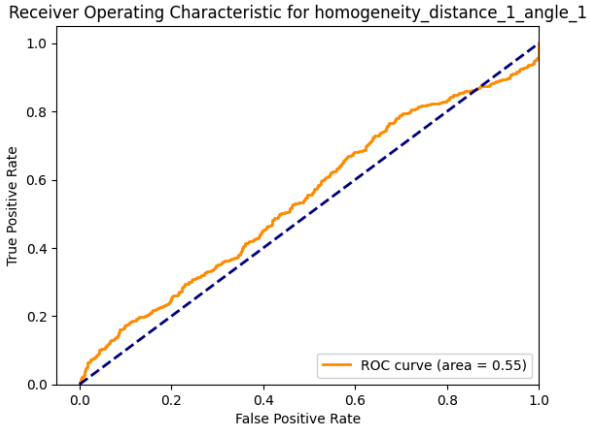
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.79 | 0.78 | 740 |
| 1 | 0.80 | 0.77 | 0.78 | 787 |
| accuracy | | | 0.78 | 1527 |
| macro avg | 0.78 | 0.78 | 0.78 | 1527 |
| weighted avg | 0.78 | 0.78 | 0.78 | 1527 |

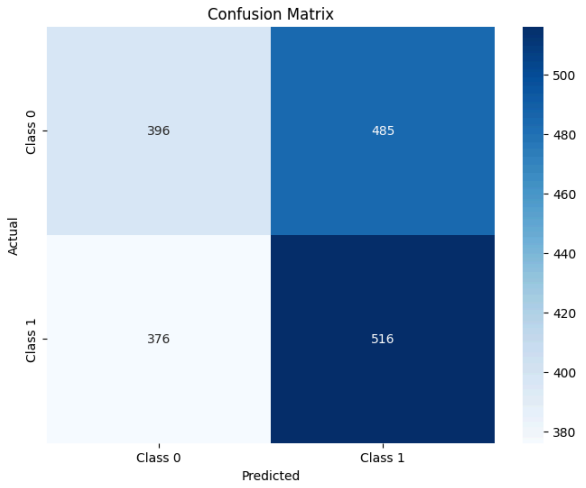
Feature importance analysis allowed visualization of relative feature importance. The performance of the logistic regression model using the most important feature was considered poor.

*Full Feature Set - Best Feature*




```
Best Feature: homogeneity_distance_1_angle_1
Best Feature Column Number: 85
Coefficient Value: -3.633496568156197
```

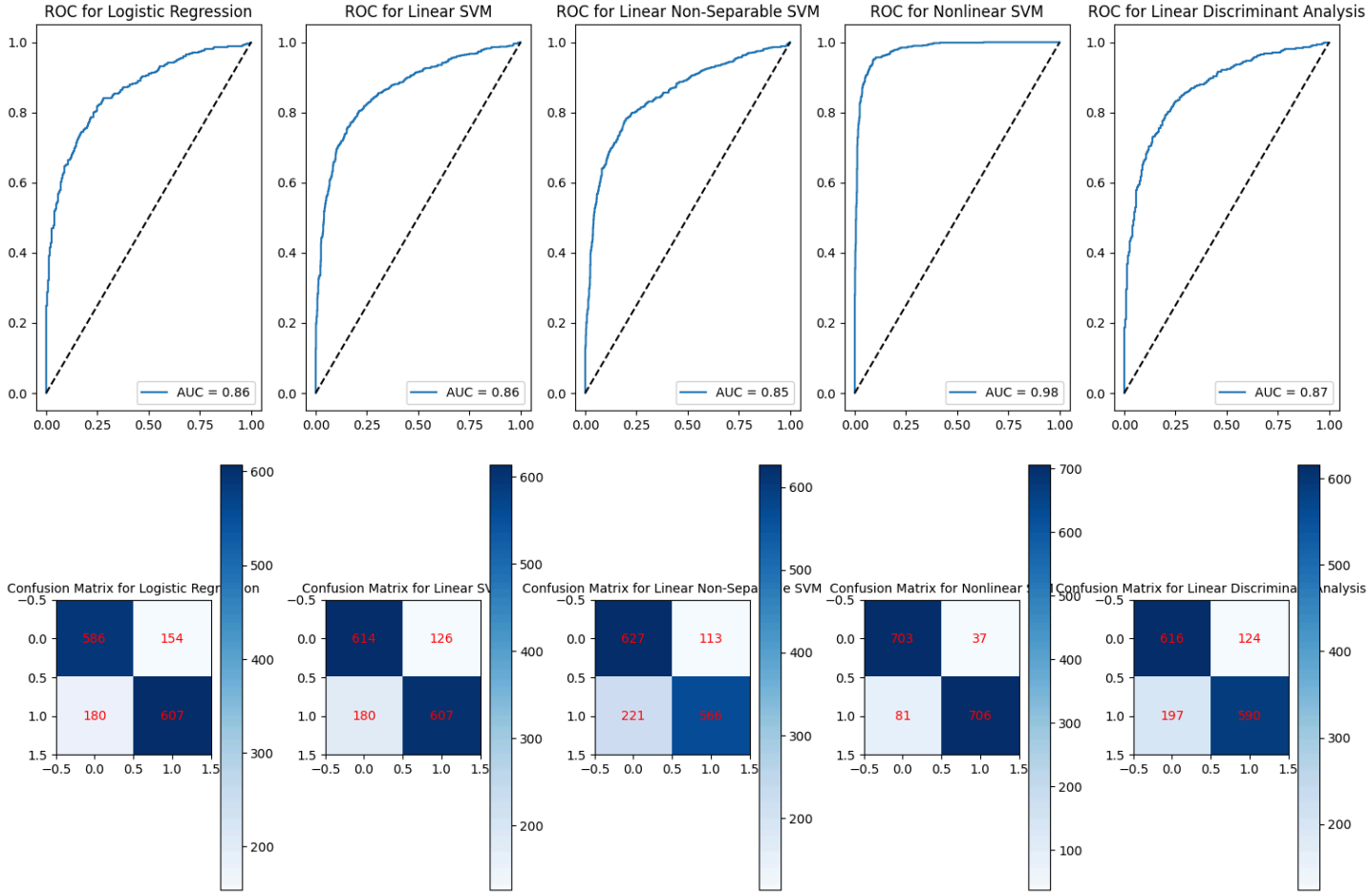


**Classification - Classifier Comparison**

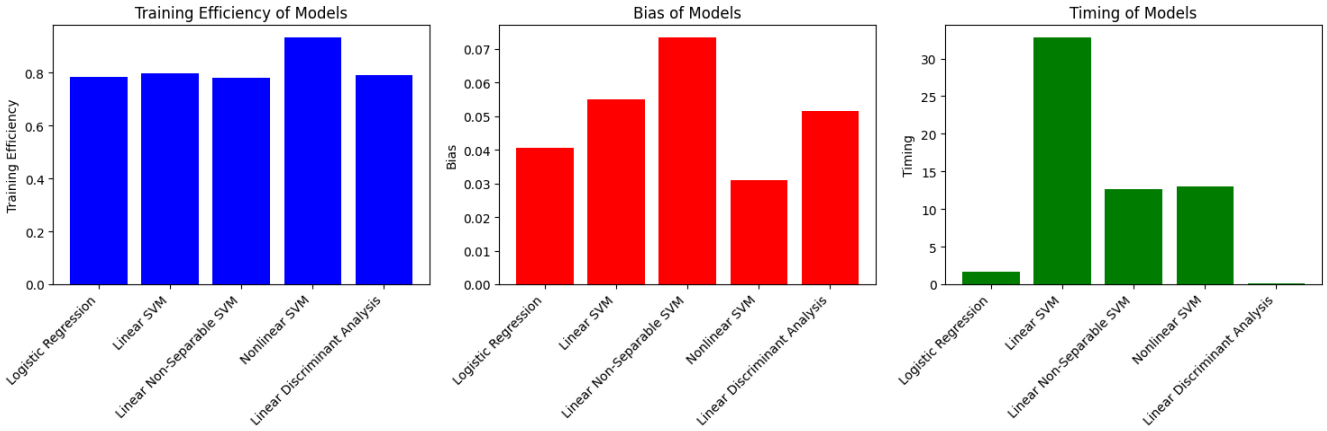
Comparing the baseline classifier to alternative linear and nonlinear classifiers identified nonlinear SVM as the best classifier.

*Full Feature Set*

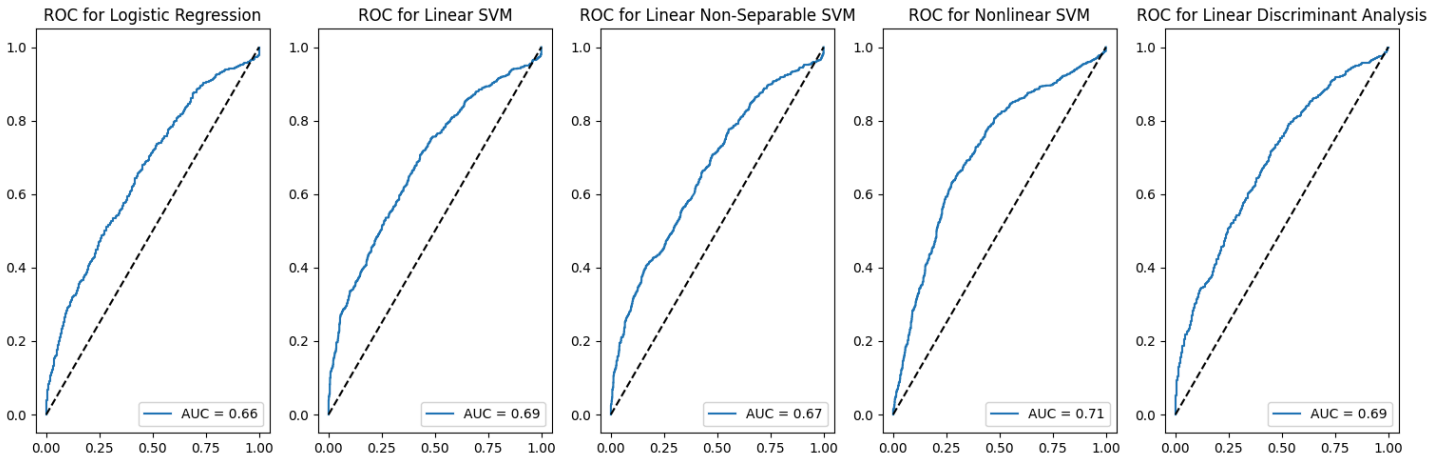


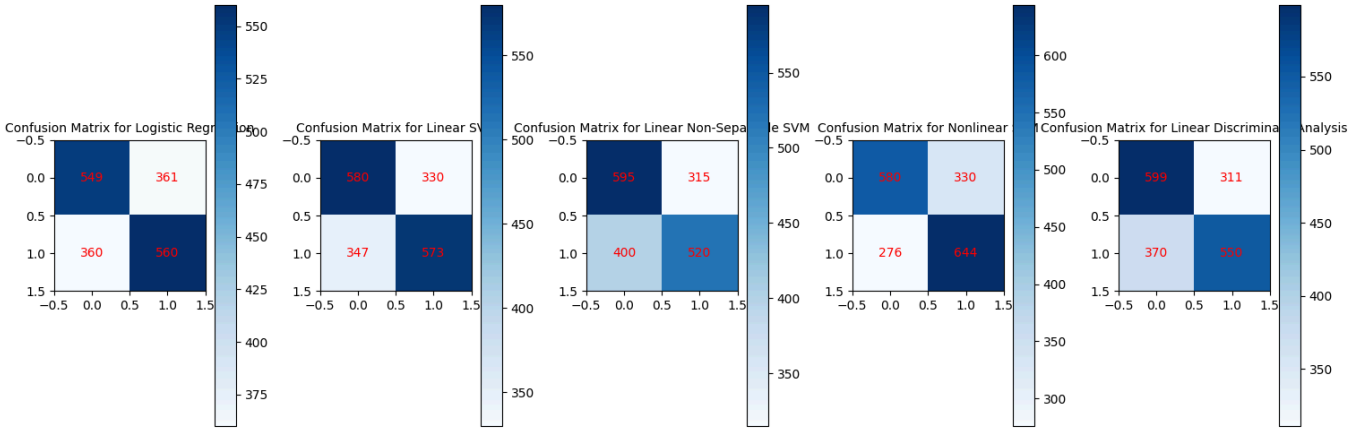

| | Accuracy | Recall | F1 Score | AUC | Training Efficiency | Bias | Timing |
|---|---|---|---|---|---|---|---|
| Logistic Regression | 0.781270 | 0.771283 | 0.784238 | 0.860651 | 0.784638 | 0.040452 | 1.668898 |
| Linear SVM | 0.799607 | 0.771283 | 0.798684 | 0.864437 | 0.796594 | 0.055028 | 32.794093 |
| Linear Non-Separable SVM | 0.781270 | 0.719187 | 0.772169 | 0.848552 | 0.782509 | 0.073370 | 12.603667 |
| Nonlinear SVM | 0.922724 | 0.897078 | 0.922876 | 0.977302 | 0.933672 | 0.030953 | 12.957023 |
| Linear Discriminant Analysis | 0.789784 | 0.749682 | 0.786143 | 0.867202 | 0.792827 | 0.051589 | 0.122248 |

Nonlinear SVM demonstrated the highest training efficiency and the lowest bias, but was slower to train compared to logistic regression and LDA.



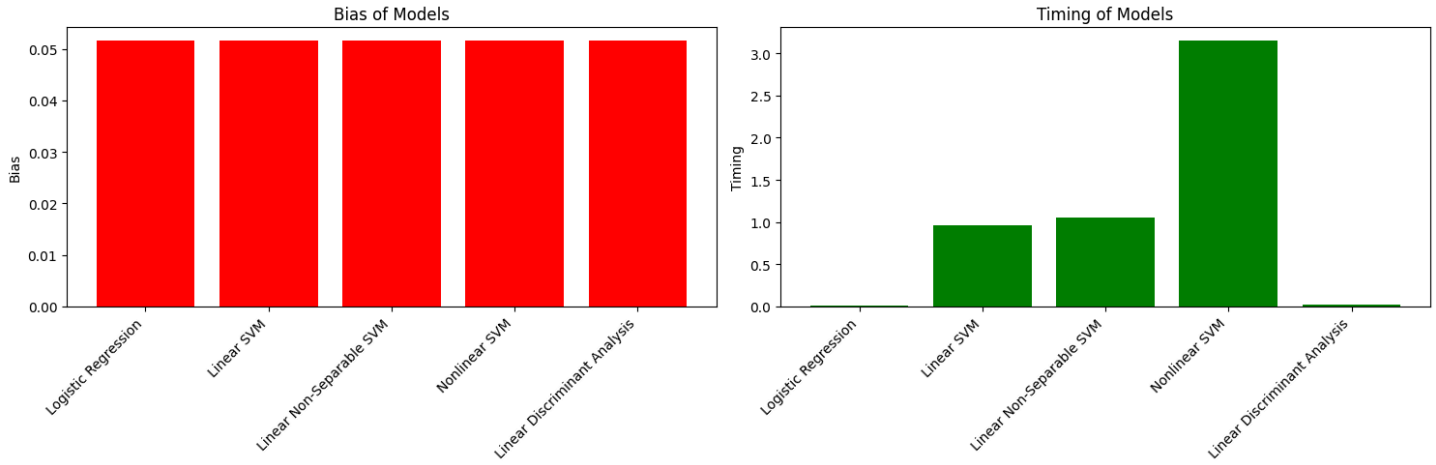

Testing nonlinear SVM model on the validation set (previously unseen data) yielded only fair model performance.



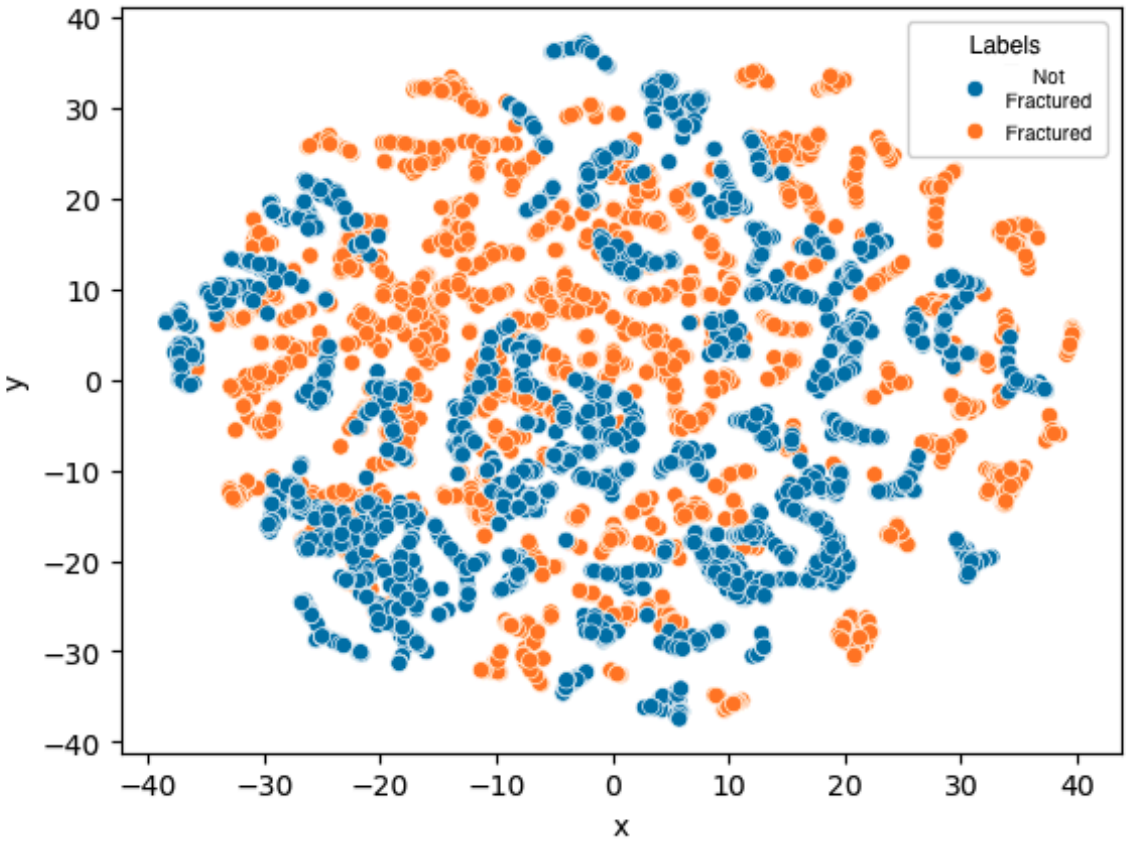

| | Accuracy | Recall | F1 Score | AUC | Bias | Timing |
|---|---|---|---|---|---|---|
| Logistic Regression | 0.606011 | 0.608696 | 0.608365 | 0.659829 | 0.051589 | 0.007922 |
| Linear SVM | 0.630055 | 0.622826 | 0.628634 | 0.686000 | 0.051589 | 0.961154 |
| Linear Non-Separable SVM | 0.609290 | 0.565217 | 0.592593 | 0.665195 | 0.051589 | 1.055426 |
| Nonlinear SVM | 0.668852 | 0.700000 | 0.680042 | 0.712011 | 0.051589 | 3.148629 |
| Linear Discriminant Analysis | 0.627869 | 0.597826 | 0.617631 | 0.686075 | 0.051589 | 0.020060 |

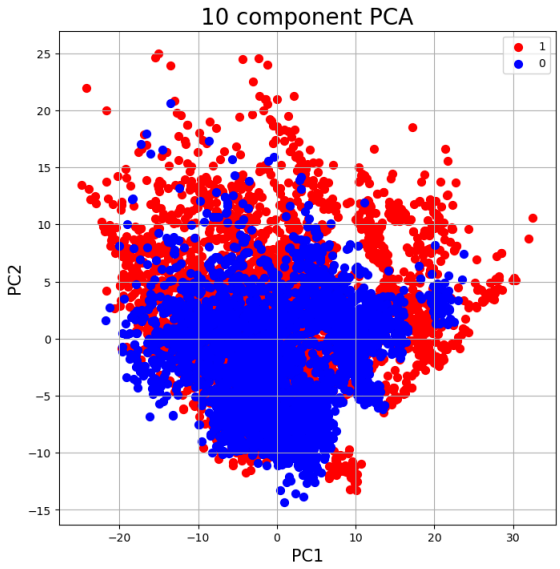
On the validation set the nonlinear SVM model had the longest training time.

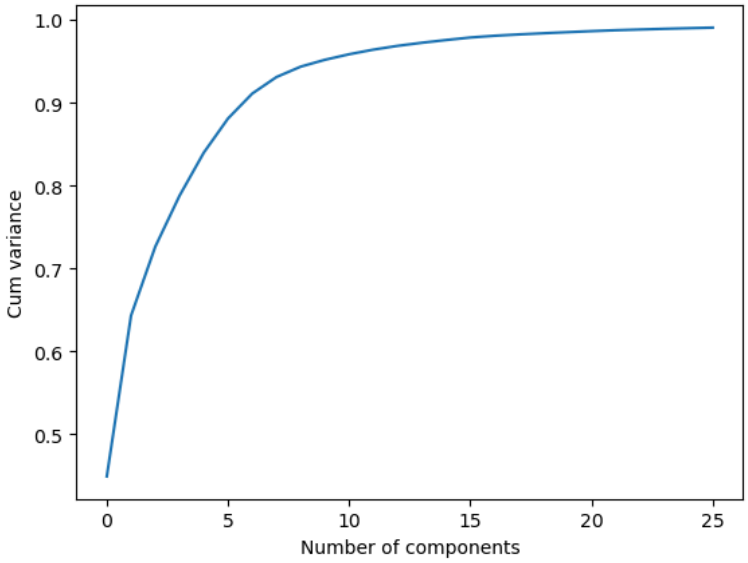



The t-distributed Stochastic Neighbor Embedding (TSNE) visualization of our data was particularly effective in illustrating the non-linear separability of our training data. This aligns well with the higher performance observed in Non-linear SVM models. The visualization distinctly displays fractures in orange and normal X-rays in blue. There are some clusters that we still not separable and possibly smaller or less represented fractures (e.g., wrists vs. legs vs. arms).

*Visualizations TSNE and PCA Feature Set*






PCA served as a useful tool in reducing the dimensionality of our data from 174 features to just 26, while retaining 99% of the variance. While there is considerable overlap observed in the first two principal components, the distribution of the fractures in red is much greater than that of the normal X-rays in blue, again suggesting the use of Non-linear models.

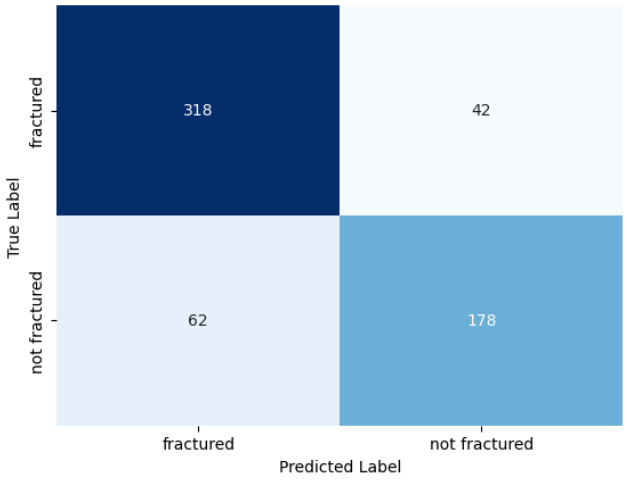
**Classification - CNN, Neural Network, & ResNet-50 Models**

To test the robustness of the ResNet-50 model and its features, we employed four models:

1. A CNN with extracted ResNet-50 features
2. A neural network with our normalized LBP features
3. A CNN with both extracted ResNet-50 features and our normalized LBP features
4. The ResNet-50 CNN with trainable layers

The layers were selected based on multiple manual tests that emulated automated hyperparameter testing. Most hyperparameter testing for our Neural Networks and CNN's were challenging due to only having a 12GB RAM machine available to us. For future testing, we plan to perform more robust hyperparameter testing on our Neural Networks and CNN's. In general, we found that models including the ResNet features required convolutional layers on top, as well as dropout and/or max pooling layers for max pooling. We also found that our LBP features alone generalized the best with one hidden layer, and our ResNet-50 CNN model with trainable layers did not require any additional layers.

As seen in *Table I*, the ResNet-50 CNN with trainable layers produced the highest validation accuracy at 83% among the four models, though on an overfit model with a training accuracy of 99%. Based on the confusion matrix below, the model had a high true positive rate of predicting fractures at 88%. The false positive rate of 26% was more concerning, but overall there was low class imbalance in prediction accuracy.




The most generalizable model relative to training was the CNN with ResNet and LBP features, which had a 79% validation accuracy compared to 80% for training. For the CNN's and NN's without ResNet pre-training, the LBP features were more predictive and generalizable than the extracted ResNet features. Even with just a dropout layer of rate 0.7 on top of logistic regression, the LBP features alone were able to achieve a validation accuracy of 69% compared to 68% training accuracy, whereas the CNN with ResNet features only predicted a value of "not fractured" for all validation images. For further testing, without time constraints, we will consider applying PCA and even more overfitting-prevention than a dropout layer of rate 0.7, given that the flattened ResNet-50 output is 100,352 features.

**Table I. ResNet, NN, & CNN Results**

| **Model** | **Layers** | **Train Accuracy** | **Validation Accuracy** |
|---|---|---|---|
| CNN w/ ResNet50 Feature | • Conv2D layer of 12 units, 3x3 kernels<br>• MaxPooling2D layer, 3x3<br>• Dropout layer, rate 0.7 | 97% | 48%, only predicted not fractured |
| Logistic Regression w/ Dropout & LBP Normalized Features | • Dropout layer, rate 0.7 | 68% | 69% |
| CNN w/ LBP Normalized Features & ResNet50 Feature | • Hidden layer of 16 units (LBP Features)<br>• Conv2D layer of 16 units, 3x3 kernels (ResNet50 features)<br>• MaxPooling2D layer, 3x3 (ResNet50 features)<br>• Dropout layer, rate 0.7 (LBP features) | 80% | 79% |

| | | | |
|---|---|---|---|
| | ● Dropout layer, rate 0.95 (ResNet50 features) | | |
| ResNet50 CNN | No additional layers | 99% | 83% |

**Generalizability**

The original source dataset has images split as:

- Fractured: 4,840 images (4,480 training, 360 validation)
- Normal: 4,623 images (4,383 training, 240 validation)

In order to make sure we had a good testing set with an 80:20 split of training to test data. We randomly sampled (blindly) a balanced amount of X-rays from the downloaded training set to add to the downloaded validation set, creating our test set. All original and rotated X-ray images were moved over to the test set, if one was selected from the training set all images pertaining to that X-ray were moved to ensure no leakage of information to the training set. With these steps performed, we ended with balanced sets of 80% training (7,633 images), 20% testing (1,830 images). To support model optimization and hyperparameter tuning, we further split the 7,633 training images into 80% training, 20% validation.

Below is a summary table of the data pipeline used, its main components and associated parameters, along with these parameters' corresponding initial values. From there we list a series of basic hyperparameter adjustments to gain intuition on model tuning. (Note this is in addition to the ResNet-specific tuning described in the preceding section.) This exercise aims to uncover the impact of i) removing pre-processing steps or features altogether, ii) compressing features via principal components analysis, and iii) tweaking core feature and model parameters outside of their default values.

While this hyperparameter search is not exhaustive, it provides cues toward each component's relative sensitivity in the current data pipeline. Namely, the model's performance sensitivity to image preprocessing and feature post-processing. No preprocessing combined with GLCM and LBP features yields a high validation AUC of 0.977 against a training AUC of 0.934, indicating a strong model generalization without significant overfitting. Preprocessing variants include the absence of the Gaussian Filter, Contrast Enhancement, and feature extraction methods, which all resulted in lower training and validation AUCs, demonstrating that image preprocessing steps, as implemented, remove critical information for accurate model predictions.

Post-processing of our best feature extraction using Principal Component Analysis (PCA) showed no improvement to our AUC. This implies that the dimensionality reduction from 140 features to 26 PCA components does not improve our best model the Non-linear SVM. With our best model identified we proceeded to evaluate its generalizability using the testing data. Despite the model performing well with a high validation AUC of 0.98, its performance significantly declined when applied to the test data, with an AUC of 0.71. This suggests that contrary to the validation performance, the model may have overfitted or not been trained on enough data to capture the different patterns observed in our testing data.

**Summary of Data Pipeline Parameters**

| # | Component | Parameter | Base Values |
|---|---|---|---|
| 1 | Image pre-processing | Resize | Resize to 224 x 224 |
| | | Gaussian blur | Kernel size = (5, 5), SigmaX = 0.55, SigmaY = 0.55, borderType |
| | | Threshold for contrast limit | clipLimit = 3 |
| | | Edge detection | Threshold1 = 50, Threshold2 = 150 |
| 2 | Feature 1: GLCM | Distances | [1, 3, 5, 9] |
| | | Angles | [0, np.pi / 4, np.pi / 2, 3 * np.pi / 4, 5 * np.pi / 4, 3 * np.pi / 2, 7 * np.pi / 4] |
| | | Properties | ['energy', 'correlation', 'dissimilarity', 'homogeneity', 'contrast'] |
| 3 | Feature 2: LBP | Radius | 2 |
| | | Number of points | 16 * radius |
| | | Method | Uniform |
| | | Bin size | Number of points + 3 |
| | | Number of bins | Number of points + 2 |
| | | Histogram normalization | Divide by mean |
| 4 | Feature 3: ResNet | Kernel | 1D |
| | | Layers / Hidden layers | No additional layers - just logistic regression |
| | | Activation | Relu |
| | | Batch Size | 128 |
| | | Optimizer | Adam |
| | | Epochs | 10 |
| 5 | Feature post-processing | Scale | Use Standard Scaler |
| | | PCA | [Number of Components] |
| 6 | Model 1: Logistic Regression | - | - |
| | Model 2: Linear SVM | Kernel | Linear |
| | Model 3: Linear non-Seperable SVM | Kernel | Linear |
| | | C | 0.1 |
| | Model 4: Non-linear SVM | Kernel | rbf |
| | Model 5: Linear Discriminant Analysis | - | - |

**Hyperparameter Testing on Non-linear SVM**

| # | Component | Change Applied | Training Area Under the Curve | Validation Area Under the Curve |
|---|---|---|---|---|
| 1 | Image pre-processing | No Gaussian Filter, with GLCM and LBP Features | 0.736 | 0.799 |
| 2 | Image pre-processing | No Contrast Enhancement, with GLCM and LBP Features | 0.720 | 0.794 |
| 3 | Image pre-processing | No Contrast Enhancement or Gaussian Filter, with GLCM and LBP Features | 0.748 | 0.800 |
| 4 | Image pre-processing | No Preprocessing only GLCM Features | 0.890 | 0.948 |
| 5 | Image pre-processing | No Preprocessing only LBP Features | 0.893 | 0.945 |

| | | | | |
|---|---|---|---|---|
| 6 | Image pre-processing | No pre-processing, with GLCM and LBP Features | 0.934 | 0.977 |
| 7 | Feature post-processing | No pre-processing, with only 26 PCA components | 0.934 | 0.977 |
| 8 | Generalization | Best Model Inference on Testing Data | | 0.710 |

**Efficiency and Accuracy**

| Model Type | Original Training Time | Original AUC | PCA Training Time | PCA AUC | Difference in Timing | Difference in AUC |
|---|---|---|---|---|---|---|
| Logistic Regression | 0.545323 | 0.860615 | 0.468648 | 0.860615 | -14.06% | 0.00% |
| Linear SVM | 22.555817 | 0.864453 | 22.464965 | 0.864453 | -0.40% | 0.00% |
| Linear Non-Separable SVM | 13.775413 | 0.848532 | 13.90952 | 0.848532 | 0.97% | 0.00% |
| Nonlinear SVM | 13.16689 | 0.977303 | 13.404149 | 0.977303 | 1.80% | 0.00% |
| Linear Discriminant Analysis | 0.782068 | 0.867202 | 0.143712 | 0.867202 | -81.62% | 0.00% |

The above table lists the training times of our various models on our best dataset (GLCM and LBP features with no preprocessing) and the set of 26 principal component features found using PCA that explained 99% of the variance in our data. Efficiency and Accuracy are important metrics to consider in modeling especially as your dataset increases in size. We focused on comparing the training times for our various models and any impacts on our models' Area Under the Curve scores. The implementation of PCA generally improved training efficiency without compromising model performance. Specifically, models with linear decision boundaries, such as logistic Regression and linear discriminant analysis, showed significant reductions in training time, while SVM models showed nearly no difference in training times or accuracy.

**DISCUSSION**

Main Findings

The best-performing model was the Non-linear SVM, which achieved an AUC of 0.97. However, its generalization to test data was moderate, with an AUC of 0.71, indicating a fair performance. Non-linear SVMs appear to be particularly suited for this task due to their capability to handle the non-linearities observed in the medical image data, providing a more flexible decision boundary compared to Linear SVMs and Logistic Regression. Using PCA reduced the number of features to 26, which maintained the ROC performance while simultaneously reducing training time in some models. Although the ResNet50 CNN showed promising results on the validation set, it did not surpass the performance of the

Non-linear SVM. Finally, the model could not be used effectively in its current form to create a bounding box to target the exact location of fractures.

Limitations

Some of the main limitations identified within our framework were:

- Test-Train split was done randomly, which may have caused imbalances in the data sets.
- A majority of the images were rotation views of the same fracture.
- Normal X-rays had more rotation views than fracture X-rays.
- Images may have undergone unknown pre-processing steps.

Future Directions

To enhance the performance of these models, it is advisable to add more independent images to the dataset and reduce the amount of augmentation currently being used. Additionally, relabelling the dataset, particularly focusing on smaller sections of the images, may aid in the development of the bounding box algorithm. This approach could provide more precise data for training and improve the algorithm's accuracy in identifying specific features within the images. More pointedly, one avenue which was suggested during in-class discussions is to i) bundle all images which pertain to a same individual (patient) and ii) keep only instances where the model is able to predict with strongest confidence. This would entail manual relabelling of the dataset, which is time-consuming and error-prone given the lack of meta-data available. It therefore extends beyond the scope of this final project but should be considered for further advancement.

**REFERENCES**

1. Cauley JA. The global burden of fractures. Lancet Healthy Longev. 2021 Sep;2(9):e535-e536.
2. Thomas JL. WHAT INFLUENCE HAS THE USE OF X RAYS HAD UPON TREATMENT OF FRACTURES AND DISLOCATIONS? Br Med J. 1906 May 5;1(2366):1034-7.
3. Kahn CE Jr, Carrino JA, Flynn MJ, Peck DJ, Horii SC. DICOM and radiology: past, present, and future. J Am Coll Radiol. 2007 Sep;4(9):652-7.
4. Kalmet PHS, Sanduleanu S, Primakov S, Wu G, Jochems A, Refaee T, Ibrahim A, Hulst LV, Lambin P, Poeze M. Deep learning in fracture detection: a narrative review. Acta Orthop. 2020 Apr;91(2):215-220.
5. Pinto A, Berritto D, Russo A, Riccitiello F, Caruso M, Belfiore MP, Papapietro VR, Carotti M, Pinto F, Giovagnoni A, Romano L, Grassi R. Traumatic fractures in adults: missed diagnosis on plain radiographs in the Emergency Department. Acta Biomed. 2018 Jan 19;89(1-S):111-123.
6. Kosrat Dlshad Ahmed, Roojwan Hawezi. Detection of bone fracture based on machine learning techniques. Measurement: Sensors, Volume 27, June 2023, 100723. Science Direct. Consulted on April 1, 2024 at https://www.sciencedirect.com/science/article/pii/S2665917423000594?via%3Dihub
7. Sebastian Hegenbart, Andreas Uhl. A scale- and orientation-adaptive extension of Local Binary Patterns for texture classification. Pattern Recognition, Volume 48, Issue 8, August 2015, Pages 2633-2644. Science Direct. Consulted April 1, 2024 at https://www.sciencedirect.com/science/article/pii/S0031320315000849
8. Bone Fracture Detection Using X-Rays, Dataset sourced from Kaggle on March 15, 2024 at: https://www.kaggle.com/datasets/vuppalaadithyasairam/bone-fracture-detection-using-xrays/data