Project Report:-

1. Data Exploration and Analysis

The dataset contains grayscale images of QR codes, categorized as "first_print" and "second_print." Visual inspection reveals subtle differences. First prints tend to have sharper edges and more consistent contrast, while second prints often show blurring, pixelation, and variations in ink density, suggesting print degradation.

- **Feature Extraction and Visualization:**
  - We can extract features like:
    - **Edge sharpness:** Using Sobel or Laplacian filters to quantify edge clarity.
    - **Contrast:** Calculating the standard deviation of pixel intensities.
    - **Blurriness:** Applying a Laplacian variance filter.
    - **Pixel intensity distribution:** Generating histograms to show intensity variations.
  - Visualizations include:
    - Histograms of extracted features, showing distributions for each class.
    - Sample images highlighting edge detection results.
    - Heatmaps of local variance.
- **Dataset Statistics:**
  - The code loads images, resizes them to 224x224 pixels, and labels them.
  - It reports the total number of images and the distribution of "first_print" and "second_print" labels.
  - The label encoder converts the string labels to numerical representations.

2. Feature Engineering

To enhance classification accuracy, we can engineer features that emphasize the differences observed:

- **Print Artifacts:**
    - **Frequency domain analysis (Fourier Transform):** Identifying high-frequency noise indicative of print degradation.
    - **Local Binary Patterns (LBP):** Capturing local texture variations, sensitive to print artifacts.
- **Resolution Differences:**
    - **Image gradients:** Analyzing the magnitude and direction of pixel intensity changes.
    - **Texture analysis(Gabor filters):** Capturing the texture differences that are caused by resolution differences.
- **CDP (Copy Detection Patterns) Degradation:**
    - Since the dataset is based on the first and second prints, the degradation of the patterns inside the QR code is what is being detected. To capture this, the use of local analysis of small sections inside the QR code would be beneficial.
    - A sliding window method could be used to calculate the variance of pixel values inside the window. This would allow the detection of changes in pixel density in small local areas of the QR code.

3. Model Development

Two approaches are implemented:

- **Deep Learning (CNN):**

- A Convolutional Neural Network (CNN) is designed, optimized for the Metal GPU.
- The CNN architecture includes convolutional layers, batch normalization, max-pooling, and dense layers.
- Data augmentation (rotation, shifts, flips) is used to improve robustness.
- The model is compiled with the Adam optimizer and binary cross-entropy loss.
- Mixed precision is used to improve performance on the Metal GPU.

- **Traditional Computer Vision and Machine Learning:**
    - Extracting the engineered features (edge sharpness, blurriness, LBP, etc.) from the images.
    - Using a Support Vector Machine (SVM) or Random Forest classifier.
    - Training and evaluating the model using cross-validation.
    - This traditional approach was not included in the provided code, but is part of the request.

4. Evaluation and Results

- **Metrics:**
    - The CNN model's performance is evaluated using accuracy, precision, recall, F1-score, and a confusion matrix.
    - The loss and accuracy graphs over the training epochs are also displayed.
- **Misclassification Analysis:**
    - Analyzing misclassified images to understand common errors.
    - Identifying patterns in challenging cases (e.g., extremely blurry images, low contrast).
- **Comparison:**
    - Comparing the performance of the CNN and traditional approaches.

  - Discussing the advantages and disadvantages of each method.
  - The CNN approach provided in the code shows high accuracy, but the traditional approach could be more efficient in cases where computational resources are limited.

5. Deployment Considerations

- **Computational Efficiency:**
  - The CNN model can be optimized for inference speed using techniques like quantization and model pruning.
  - The traditional approach might be more efficient for resource-constrained devices.
- **Robustness:**
  - The model should be tested with images captured under various scanning conditions (lighting, angles, etc.).
  - Data augmentation during training helps improve robustness.
- **Security Implications:**
  - The model should be protected from adversarial attacks.
  - Consider implementing security measures to prevent unauthorized access to the model.
- **Real-World Deployment:**
  - The model could be integrated into a mobile app or web service.
  - Images could be captured using a smartphone camera or a dedicated scanner.
  - The model would then make a prediction and provide an authentication result.
- The model can be deployed on a server that recieves images, and returns the result of the prediction. This server can be accessed through a web api.