Breast Cancer Wisconsin (Diagnostic)

Data Preprocessing

- Data Loading
  - The dataset is downloaded from the Kaggle dataset repository and loaded into a Pandas DataFrame.
  - Proper column names are assigned to the DataFrame to enhance readability and accessibility.
- Exploratory Data Analysis (EDA)
  - Basic EDA is performed to understand the dataset's characteristics, including displaying the first few rows, checking the shape of the dataset, and generating descriptive statistics.
  - This step is crucial for identifying the distribution of the data, spotting any anomalies or outliers, and deciding on the next steps for preprocessing.
- Handling Missing Values
  - A thorough check for missing values is conducted to ensure the integrity and completeness of the data.
  - If any missing values were found, appropriate measures would be taken to address them, such as imputing the missing values or dropping the affected rows/columns.
- Visualization
  - Visualizations are created for each feature to identify outliers and understand the data distribution. Boxplots are particularly useful for spotting outliers.
  - Scatter plots with regression lines could be generated to visualize linear relationships between each feature and the target variable, helping to understand how changes in the features might affect the target.
- Feature Scaling
  - The features are standardized using StandardScaler, ensuring they all have a mean of 0 and a standard deviation of 1.
  - Neural networks are sensitive to the scale of input data, and standardizing the features helps in accelerating the training process and achieving better model performance.

- Data Splitting
    - The dataset is split into features (X) and the target variable (y).
    - It is further divided into training (60%), validation (20%), and test sets (20%). The training set is used to train the model, the validation set is for tuning hyperparameters and to prevent overfitting, and the test set evaluates the model's performance on unseen data.
    - The data is converted to PyTorch tensors, making it compatible with the neural network training process.

Model Architecture

- The neural network, named Net, is implemented as a subclass of PyTorch's nn.Module. It is a feedforward neural network consisting of three fully connected (linear) layers, interspersed with activation functions and dropout for regularization.
- Input Layer (fc1):
    - Type: Fully Connected (Linear)
    - Size: n_features × 128
    - Activation Function: ReLU
    - Description: This layer takes in the input features (size: n_features) and transforms them to a 128-dimensional space. The ReLU activation function introduces non-linearity, allowing the network to capture complex patterns in the data.
    - Regularization: Dropout with a rate of 0.3 is applied after the ReLU activation to prevent overfitting.
- Hidden Layer (fc2):
    - Type: Fully Connected (Linear)
    - Size: 128 × 64
    - Activation Function: ReLU
    - Description: The output from the first layer is further transformed to a 64-dimensional space. ReLU is used for introducing non-linearity.
    - Regularization: Dropout with a rate of 0.3 is applied after the ReLU activation.

- Output Layer (fc3):
    - Type: Fully Connected (Linear)
    - Size: 64 × 1
    - Activation Function: Sigmoid
    - Description: This layer reduces the dimensionality from 64 to 1, producing a single scalar value as output. The sigmoid activation function ensures that the output is in the range [0, 1], which is suitable for binary classification tasks.
- Forward Pass
    - The forward method defines the sequence of operations for the forward pass of the network.
    - The input data is passed through the three layers sequentially, with ReLU activations and dropout applied after the first and second layers.
    - A sigmoid activation is applied after the output layer to produce the final prediction.

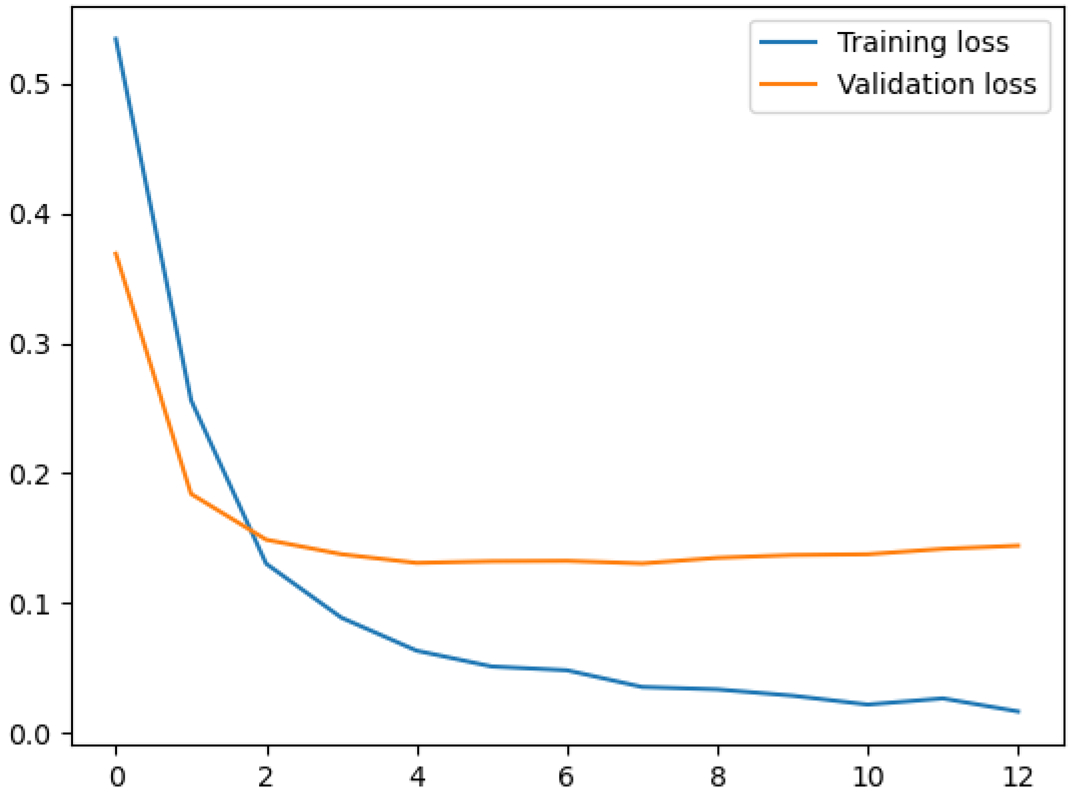
Choice of Hyperparameters

- Learning Rate
    - Value: 0.001
    - Significance: The learning rate is a crucial hyperparameter that controls the size of the steps that the optimizer takes while updating the model's weights. A too high learning rate can cause the model to converge quickly and potentially overshoot the minimum cost, resulting in an unstable training process. On the other hand, a too low learning rate can lead to a very slow convergence, or the model might not converge at all.
    - Choice: A value of 0.001 is selected for this model, which is a common choice for many problems and is often a good starting point. However, the optimal learning rate can vary depending on the specific characteristics of the dataset and problem at hand, and it might require careful tuning through experiments or techniques like learning rate scheduling.

- Number of Epochs
    - Value: 25
    - Significance: The number of epochs determines how many times the learning algorithm will work through the entire training dataset. A smaller number of epochs might not be sufficient for the model to learn the underlying patterns in the data, leading to underfitting. Conversely, too many epochs can lead to overfitting, especially if the model is complex and the dataset is not large enough.
    - Choice: The model is trained for 25 epochs, which provides a balance between giving the model enough time to learn and preventing overfitting. Early stopping is also implemented to halt the training process if the validation loss does not improve for a specified number of consecutive epochs, helping to prevent overfitting.
- Batch Size
    - Value: 16
    - Significance: The batch size controls how many training examples are used to calculate the gradient and update the model's weights at each iteration of an epoch. A smaller batch size often results in a regularizing effect and lower generalization error but can increase the time required to train the model.
    - Choice: A batch size of 16 is chosen, striking a balance between computational efficiency and the regularizing effect of smaller batch sizes. This value should provide a stable estimate of the gradient while also contributing to faster convergence.
- Activation Functions
    - Types: ReLU (hidden layers), Sigmoid (output layer)
    - Significance: Activation functions introduce non-linearities to the model, enabling it to capture complex patterns in the data. ReLU is a popular choice for hidden layers due to its simplicity and efficiency. The sigmoid function in the output layer squashes the output between 0 and 1, making the output interpretable as a probability for binary classification tasks.
    - Choice: ReLU is chosen for its advantages in training deep networks, and sigmoid is chosen for the output layer to suit the binary classification nature of the task.

- Regularization
    - Type: Dropout
    - Rate: 0.3
    - Significance: Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to 0 at each update during training. This helps in making the model more robust and prevents it from relying too much on any one feature.
    - Choice: A dropout rate of 0.3 is used, meaning that approximately 30% of the neurons are randomly dropped out during training. This value is a common starting point, but like other hyperparameters, it might require tuning based on the specific problem.
- Early Stopping
    - Description: Early stopping is employed to halt the training process when the validation loss ceases to decrease, preventing overfitting and ensuring computational efficiency.
    - Significance:
        - Balances between underfitting and overfitting.
        - Helps in choosing an optimal number of epochs, improving model generalization.
    - Implementation Details:
        - Patience: 5 epochs. Training stops if no improvement in validation loss.
        - Monitor: Validation loss.
        - Trigger: No decrease in validation loss for 5 consecutive epochs.
    - Choice:
        - A patience of 5 epochs provides a balanced approach, preventing premature stopping while ensuring efficiency and performance.

Learning Curves



- X-Axis (Epochs): Represents the number of passes through the entire training dataset. As the number of epochs increases, the model has more opportunities to learn from the data.
- Y-Axis (Loss): Represents the model's error or loss. Lower values indicate better performance.
- Training Loss: Typically decreases over time as the model learns from the training data.
- Validation Loss: Shows how well the model is performing on unseen data.
- Convergence: The training loss continues to decrease while the validation loss starts to increase, it's a sign of overfitting. This is why we need early stopping.

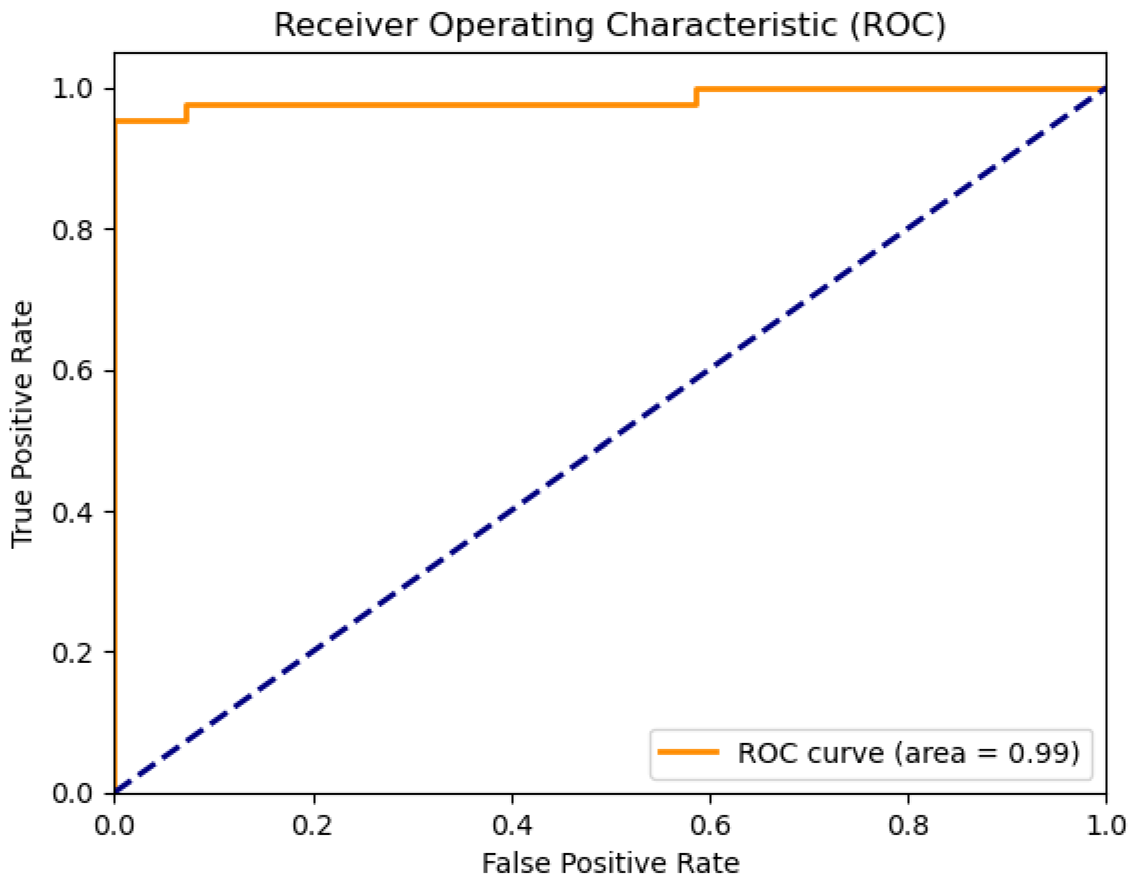
Evaluation metrics on the test set

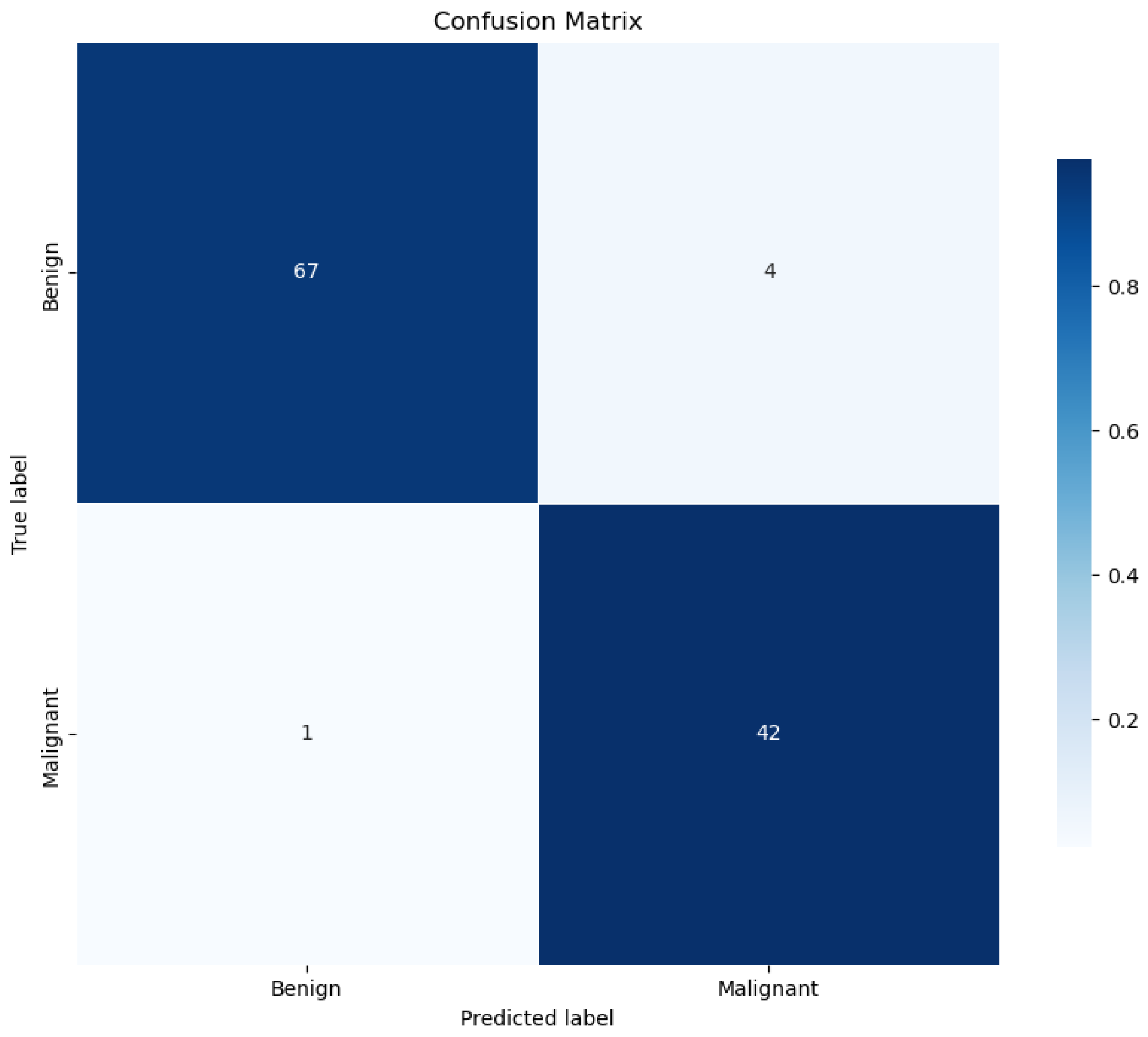
- Accuracy
  - Description: The ratio of correctly predicted observations to the total observations.
  - Interpretation: An accuracy of 1 (or 100%) indicates perfect predictions. The closer to 1, the better the model's performance.
- Confusion Matrix
  - Description: A table used to evaluate the performance of a classification algorithm. It shows the true positive, true negative, false positive, and false negative predictions.
  - Interpretation: Provides insight into the types of errors made by the model, helping to identify if it is biased towards any particular class.
- Precision
  - Description: The ratio of correctly predicted positive observations to the total predicted positives.
  - Interpretation: High precision relates to a low rate of false positives. A precision of 1 indicates that every positive prediction made by the model is correct.
- Recall (Sensitivity)
  - Description: The ratio of correctly predicted positive observations to all actual positives.
  - Interpretation: High recall relates to a low rate of false negatives. A recall of 1 means the model captured all actual positives.
- F1 Score
  - Description: The weighted average of Precision and Recall.
  - Interpretation: An F1 Score of 1 indicates a perfect balance between precision and recall. The closer to 1, the better the model’s balance between false positives and false negatives.
- AUC-ROC
  - Description: The area under the Receiver Operating Characteristic (ROC) curve, a graphical representation of the model’s ability to distinguish between classes.
  - Interpretation: An AUC-ROC of 1 indicates a perfect model; the closer to 1, the better the model is at distinguishing between positive and negative classes.

- Cross Entropy Loss
    - Description: A measure of how well the predicted probabilities match the actual class labels, being low for good models.
    - Interpretation: A Cross Entropy Loss of 0 indicates perfect predictions. The lower the loss, the better the model's performance.

Result

Accuracy: 0.956140350877193
Precision: 0.9130434782608695
Recall: 0.9767441860465116
F1 Score: 0.9438202247191011
AUC-ROC: 0.996069439895185
Cross Entropy Loss: 0.09213235229253769





Summary

- Architecture:
  - Description: The neural network structure is straightforward yet adequately complex, facilitating the model's ability to comprehend non-linear patterns within the dataset.
  - Evaluation: The model architecture, consisting of three layers with dropout for regularization, is aptly designed for the binary classification task at hand.
- Hyperparameters:
  - Description: The hyperparameters selected are conventional, aiming to strike a balance between rapid convergence and model stability.

- Learning Curves:
  - Overfitting: The learning curve indicates that the model is performing significantly better on the training data compared to the validation data.
  - Addressing Overfitting with Early Stopping:
    - Early stopping involves halting the training process once the model's performance on the validation set stops improving.
- Final Thoughts:
  - Proactive Approach: Ongoing vigilance and fine-tuning of both the network architecture and hyperparameters are paramount for optimal model performance.
  - Benchmarking: Comparing the model’s results to baseline models or industry standards can offer additional validation and insights, ensuring the reliability of the predictions.
  - Outcome: Adhering to these practices guarantees a comprehensive and effective modeling approach, culminating in precise and dependable outcomes.