Report on Results and Analysis of Classification Models

Name: Julio Alejandro Serrepe Ramírez
Date: August 31, 2025

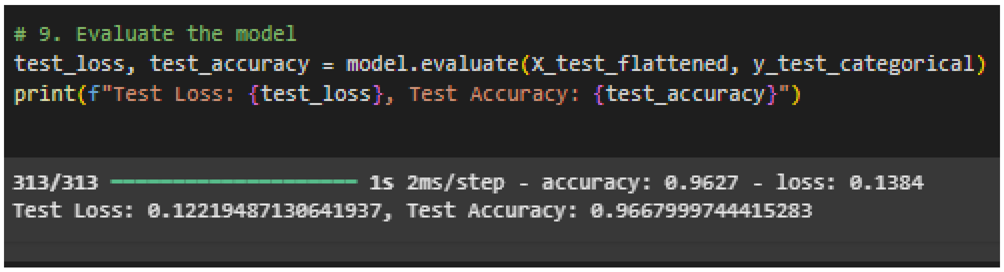
---

1. Notebook 1:

Objective: To build and train a dense (fully connected) neural network for a classification problem involving handwritten digits from the MNIST dataset.

1.1. Key Results

After running the notebook, the neural network model achieved the following results on the test dataset:



```
# 9. Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test_flattened, y_test_categorical)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9627 - loss: 0.1384
Test Loss: 0.12219487130641937, Test Accuracy: 0.9667999744415283
```

- **Final Test Accuracy:** 96.68%
- **Final Test Loss:** 0.1221

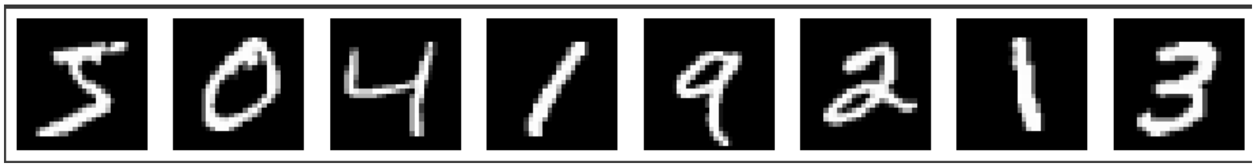
1.2. Analysis and Observations

The model's performance was notably high, achieving nearly 97% accuracy in classifying digits it had never seen before. My key observations are as follows:

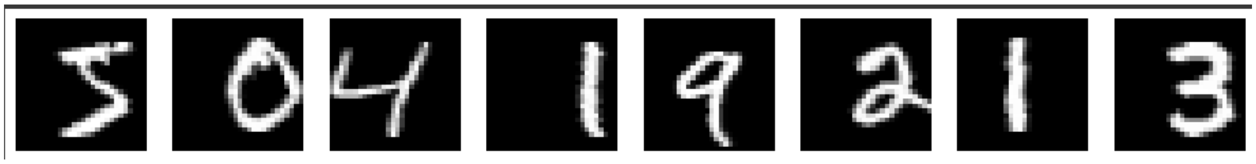
- **Importance of Pre-processing (**
  One of the most critical steps was the *deskew()* function, which corrects the slant of the digits. This step is crucial because it standardizes the images, allowing the neural network to focus on the essential features of each digit (curves, lines, intersections) rather than also having to learn to interpret variations in tilt. A comparison of the images before and after this

correction is shown below.

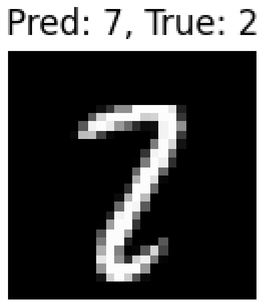
Original:



After *deskew()* function:



- **Training Performance:**
  During the training process (*model.fit*), it was observed that the training accuracy (*accuracy*) progressively increased with each of the 10 epochs, while the loss (*loss*) decreased. The validation metrics (*val_accuracy*) also improved, indicating that the model generalized well and did not show significant signs of overfitting.
- **Analysis of Model Errors:**
  The notebook visualizes some of the digits that the model classified incorrectly. Upon observing these errors, it is noticeable that many of the digits are ambiguous or atypical (e.g., a "2" that looks like a “7” or a "1" that is mistaken for a "7"). This demonstrates that although the accuracy is high, the model still struggles with edge cases that might even be challenging for a human to classify.



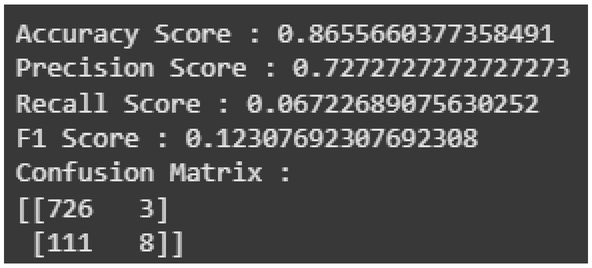
---

2. Notebook 2:

Objective: To analyze a medical dataset to predict the 10-year risk of coronary heart disease by training a logistic regression model and evaluating its performance.

2.1. Key Results

The initial logistic regression model, without hyperparameter tuning, yielded the following metrics:



```
Accuracy Score : 0.8655660377358491
Precision Score : 0.7272727272727273
Recall Score : 0.06722689075630252
F1 Score : 0.12307692307692308
Confusion Matrix :
[[726   3]
 [111   8]]
```

- **Accuracy Score:** 86.55%
- **Precision Score:** 72.72%
- **Recall Score (Sensitivity):** 6.72%
- **F1 Score:** 0.1230
- **Confusion Matrix:** [[726, 3], [111, 8]]

After optimization with *GridSearchCV*(), the accuracy was:

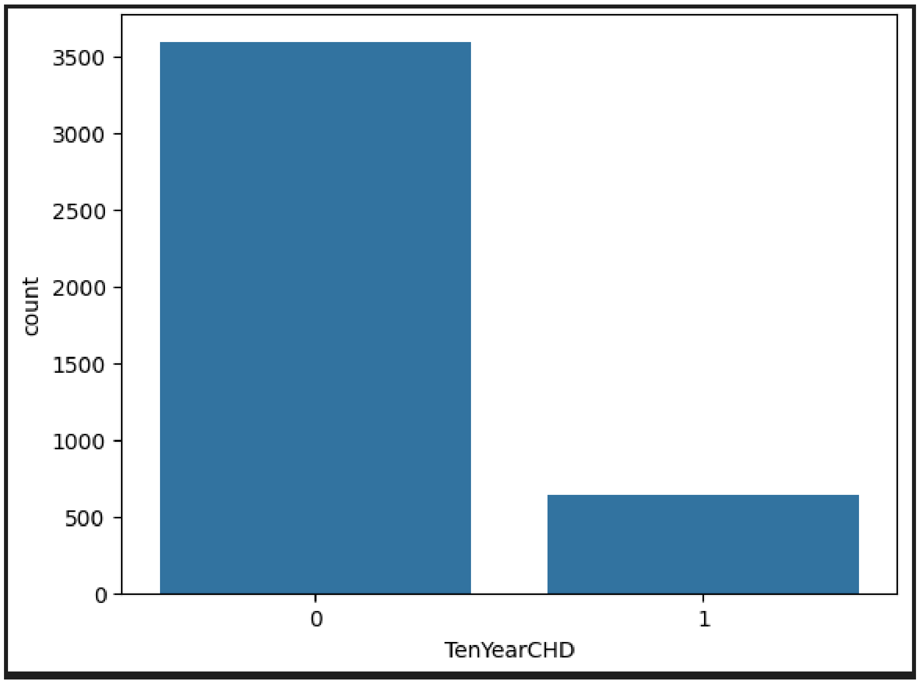
- **Accuracy after optimization:** 86.08%

2.2. Analysis and Observations

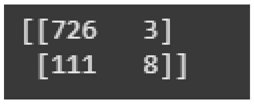
Unlike the first notebook, the results of this model must be interpreted with great care due to the nature of the problem and the data.

- **The Problem of Imbalanced Data:**
  The first major finding is the severe imbalance in the target variable (*TenYearCHD*). As seen in the plot, the vast majority of patients (value 0) did not develop the disease. This creates a

bias: a model can achieve high accuracy simply by predicting "0" for almost every case.



- **Misleading Metrics vs. Critical Metrics:**
  The initial accuracy of **86.55%** appears high but is misleading due to the data imbalance. The most revealing metric here is the **Recall (Sensitivity)**, which is extremely low (**6.7%**). This means the model was only able to correctly identify 6.7% of the patients who **DID** develop heart disease. In a medical context, this is a critical failure. The **111 "False Negatives"** (from the confusion matrix) represent at-risk patients whom the model failed to detect, which could have serious health consequences.



```
[[726   3]
 [111   8]]
```

- **Effect of Optimization and Data Cleaning:**
  Important decisions were made, such as filling missing values (e.g., *glucose*) and dropping the *education* column. These steps are essential for the model to be able to train. Interestingly, hyperparameter tuning with *GridSearchCV()* did not improve accuracy; in fact, it slightly decreased it. This suggests that the primary issue is not the model's hyperparameters but the inherent imbalance in the data.

---

3. General Learnings and Comparison

By comparing both notebooks, I learned the following:

1. **Context is Key for Metrics:** The most important evaluation metric depends on the problem. For digit classification (Notebook 1), high ***Accuracy*** is a good indicator of success. However, for disease prediction (Notebook 2), ***Recall*** is far more critical to avoid false negatives.
2. **Pre-processing is Fundamental:** Both models required careful pre-processing, but of different kinds. One standardized images (*deskew()*), while the other handled missing data in a tabular format. This shows there is no one-size-fits-all solution for data preparation.
3. **Simple vs. Complex Models:** Notebook 1 used a neural network (more complex) for a perception problem (images), whereas Notebook 2 used logistic regression (simpler and more interpretable) for tabular data. This is a common and appropriate practice for each type of problem.

---

4. Questions and Doubts

This exercise raised the following questions for me:

1. **For the Heart Notebook:** Given the low Recall, what techniques could be used to improve the detection of true positives? I have read about methods like SMOTE (to balance the data) or adjusting class weights in the model. Would these be the logical next steps?
2. **For the Digits Notebook:** The dense neural network performed well, but for image problems, Convolutional Neural Networks (CNNs) are often used. What advantage would a CNN offer for this problem compared to the fully connected network that was used?
