25.09.2025
Ken Lam and Ognian Trajanov
Professor Shawn Chen

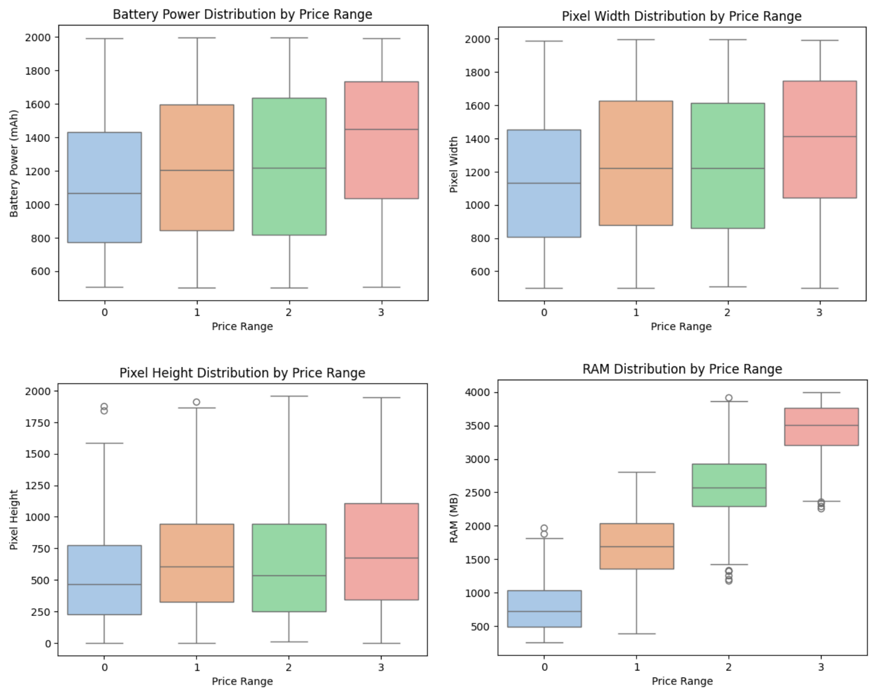
CS 366 - Assignment 1 Report

For this assignment we have built a Multi-Layer Perceptron that predicts the price range of mobile phones based on their hardware specifications.

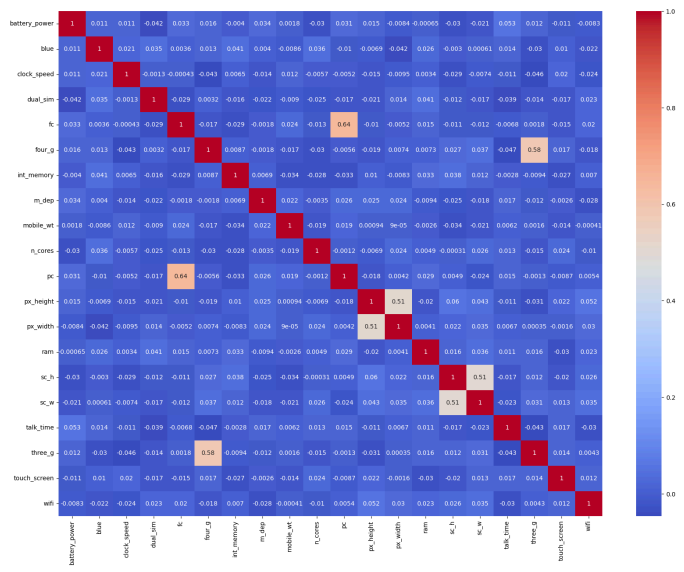
1. EDA

The first step we took was analyzing the features and labels of the dataset. Upon performing the EDA we found the following patterns:

- Relatively equal distribution of: battery power, price range, and Bluetooth capability in the dataset.
- The features that we found are most strongly correlated with the difference in price range are: battery power duration, px_height, px_width, RAM (in MB).



- We found that the Primary camera in mega pixels and the Front camera in mega pixels features were strongly correlated. Additionally, 4G and 3G status are also strongly correlated (i.e., a phone that has 4G is also compatible with 3G). Lastly, a screen's width and height are strongly correlated, which makes sense.



2. Preprocessing

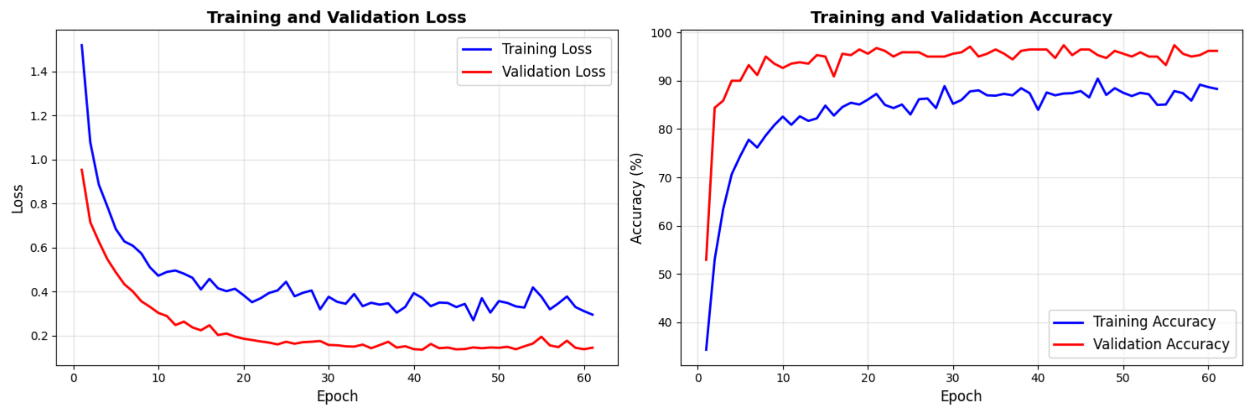
We preprocessed the dataset by first standardizing all numerical features so they follow a normal distribution with mean 0 and variance 1, ensuring comparability across scales. Then, we performed feature selection by dropping the redundant and uninformative columns whose values do not vary meaningfully across the price range. The features we dropped were namely: *three_g, touch_screen, wifi, dual_sim, blue, talk_time, pc*. These features have similar/near identical distributions across price ranges so including them in our model wouldn't be as helpful.

3. MLP

We implemented an MLP classifier with the following hidden layers (256 => 128 => 64 => 32) and an output layer of size 4. Each layer is followed by batch normalization, ReLU activation, and dropout (0.4) for regularization. For weight initialization, we initially implemented the Kaiming He initialization (suggested during class because it works better with ReLU activation), we observed better performance using the Xavier initialization, so we adopted it to stabilize training and achieve higher validation accuracy. For our optimizer, we used Adam with a weight decay of 0.0001 and a starting learning rate of 0.001. We also used a learning rate scheduler with a patience of 20 epochs and a decaying factor of 0.5. Our early stopping's patience is 20 epochs.

4. Training and validation loop

We created an 80/20 train/validation split, built loaders, and trained an MLP with CrossEntropyLoss and weight decay while tracking loss/accuracy. A scheduler lowers the LR on stagnant validation loss, and early stopping with patience saves and restores the best model weights. Finally, we plot learning curves and return the trained model along with the held-out test features and labels.



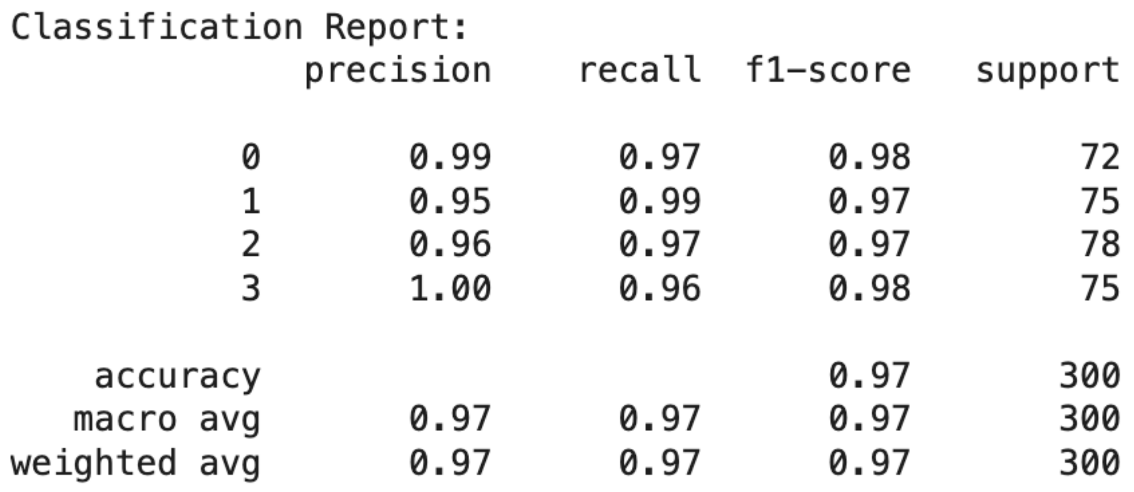
5. Final Prediction Function

The predict function tests the model on new (unseen) data. It lets the model make a guess for each example, counts how many are right to compute overall accuracy, and prints a classification report as well as a confusion matrix. Finally, it returns the accuracy.

6. Key findings:

We found that removing features that are uninformative (even distribution across all price ranges) helps to increase the model's accuracy metrics. Additionally, we found that initializing the weights with Xavier initialization contributes to a more stable loss decay.

25.09.2025
Ken Lam and Ognian Trajanov

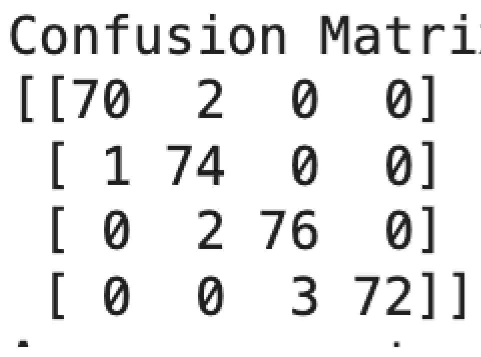


```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98        72
           1       0.95      0.99      0.97        75
           2       0.96      0.97      0.97        78
           3       1.00      0.96      0.98        75

    accuracy                           0.97       300
   macro avg       0.97      0.97      0.97       300
weighted avg       0.97      0.97      0.97       300
```

Based on the classification report, we can see that the model's precision is generally high of about 95% on average (i.e., out of the predictions for price range of 0, the model was able to get 99% of them correctly). Conversely, the recall rate for all price ranges are high, showing that most of the actual correct price ranges were captured by the model's predictions. For the F-1 score, we can see that the model was able to maximize the number of correct predictions (true positives) while minimizing the false positives and false negatives.



```
Confusion Matrix:
[[70  2  0  0]
 [ 1 74  0  0]
 [ 0  2 76  0]
 [ 0  0  3 72]]
```

Based on the confusion matrix, we can see that the model's predictions are extremely accurate with a couple of observations from the test data labeled wrong. This indicates the model's flexibility in accurately predicting unseen data.