Pierre Kachi 232732

Rami Chalouhi 221631

Said Chebbo 220700

We began by picking a dataset containing information regarding diamonds and their prices, and our goal will be to get a model to try and guess said prices. We then used pandas to read and convert it into a dictionary. Next, to visualize the data and get a better grasp of what we were working with, we utilized seaborn and matplotlib respectively to create a heatmap and a scatterplot, the first showing us the correlation between the price and the rest of the attributes, and the second to help us put into perspective all the information, and later on compare it to the generated results. We next used sklearn’s train test split to split the data into a testing and a training set, 20% for testing and 80% for training, before splitting the training set into itself and a validation set with 25% used for the latter. So in total we ended up with 60% of the data used for training, 20% for testing and 20% for validation. After picking the attributes and having partitioned the data we began building our model. We created 3 separate dense layers: the first one consists of 32 neurons, uses relu as an activation function, and we gave it an input shape parameter equal to the amount of keys found within x (the data used to try and guess y). The second layer, consists of 64 neurons, uses relu, and doesn’t take in a shape parameter as it will be using on the output of the first layer as input. The final layer only consists of one neuron, and it will be responsible for outputting the generated results. We then used model.compile and set the loss function to the mean squared error, and the metrics to the mse and the mean absolute error, with adam optimizer at 0.001. Finally we used model.fit to set the parameters of our model, telling it to use the training and validation data, which is useful in the evaluation of the model, and setting the number of epochs to 10 which we found to close enough to the convergence point without being too small. We saved the return of this function into a variable called history which was used to access the training data and find the convergence point. Our model function returns both the trained model and the history object. We found our r2 score to be around 0.86 for 10 epochs, which was satisfactory enough. Though this tool cannot be used to accurately measure a diamond’s price, it still has a use in giving the user a pretty good idea of what it should be around.