Exploratory data analysis

1. Observe your dataset

Identify its size, number of features, number of samples, are there any missing values? what would you use as your target variable

2. Analyze your data

- check if you have any missing variables
- Identify what types of features you have: categorical or numerical?
- is it a balanced dataset (i.e. how many samples for each label)
- identify ranges for numerical data, i.e. min and max values
  Note: Optimizers tend to work well with numbers in range [-1, 1].
- use **Histogram** or **Bar Chart** to visualize the distribution and variance of each variable
  Note: ideally we want to work either with normal or uniform distributions
- Locate outliers, if any
- **Identify relationships in your dataset:** Consider Pearson, Spearman or Kendall techniques.
  - look at **correlation of features with the target class label.** Features highly correlated with the target class label serve as important features. Whereas features, which are not correlated with the target class label may be not as important and can potentially be dropped.
  - look at **correlation between features.** Identify highly correlated features, if any.

3. Transform your dataset

- Deal with missing values
- Split data into train and test
- Transform categorical features into numerical (consider one-hot encoding)
- Consider doing feature scaling (min-max scaling, clipping) and centering to [-1,1] for numerical values. If you have outliers, be careful. Pick reasonable min and max values and treat outliers as -1 and 1.
- If any of your feature distributions is skewed, consider transforming it (e.g. log transformation) so that it looks more like a normal or uniform distribution
- Optional: perform feature selection

4. **Formulate your ML task**
    - Binary or non-binary classification?

5. **Think about your potential NN architecture**
    - What size of the network would be appropriate for your data. It may change during the implementation and fine-tuning part, but it is good to have an idea of where you would like to start.
    - What optimizer do you want to try?
    - What loss function will you use?
    - For how long will you train? I.e. how many epochs
    - What activation functions you want to explore: For the output layer I recommend using sigmoid for binary classification and softmax for non-binary. Consider using ReLU or its variations for hidden layers.
6. **Think what metrics will you use for your model evaluation**
    - Accuracy, precision, recall, sensitivity, specificity, AUC, F1
      Again, this may change during the implementation, but it is good to have a starting point.

Implementation with PyTorch

CoLab already has PyTorch installed.
PyTorch Quickstart: https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html

Blog post on educative.io

1. **Create custom class for your dataset, then create DataLoader**
   Refer to:
   https://androidkt.com/load-pandas-dataframe-using-dataset-and-dataloader-in-pytorch/

   https://discuss.pytorch.org/t/dataset-from-pandas-without-folder-structure/146816/4

2. Implement your NN

Define a class for your model
https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html
https://www.kaggle.com/code/ryanluoli2/beginner-guide-to-neural-networks-with-pytorch

3. Specify loss, optimizer and write training and test steps in separate functions

https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html#optimizing-the-model-parameters

4. Make your model initialization, training and any random splits deterministic

That means explicitly specify seed for numpy arrays, torch models and etc.
This is important for reproducibility

5. Split your training set into train and validation sets. As a result, you will have 3 sets: train, validation and test.

6. Train your model

https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html#optimizing-the-model-parameters

- Monitor your training via recording training and validation losses and displaying them
- If your model overfits or underfits, take care of it during step 8
- Consider saving your best model (best performing on validation data) for further evaluation

7 . Evaluate your model on the test set.

- Compute your metrics (this will be your baseline)

8. Fine-Tune/improve your model

- Consider adding regularization (L1 or L2, or Dropout), especially if you have problems with overfitting; try different learning rates for your optimizer; if your model struggles with underfitting, consider increasing your model size.
  More on regularization, overfitting and underfitting:
  https://towardsdatascience.com/techniques-for-handling-underfitting-and-overfitting-in-machine-learning-348daa2380b9
- Compare results with the baseline