Asal Rahbari – Medical Appointment

This dataset contains 14 columns, some of which are numerical and others are categorical (some are in the binary form).

There's a column for gender of the patients, some for dates of the appointment, some are related to patients' health condition like whether if someone has diabetes or not.

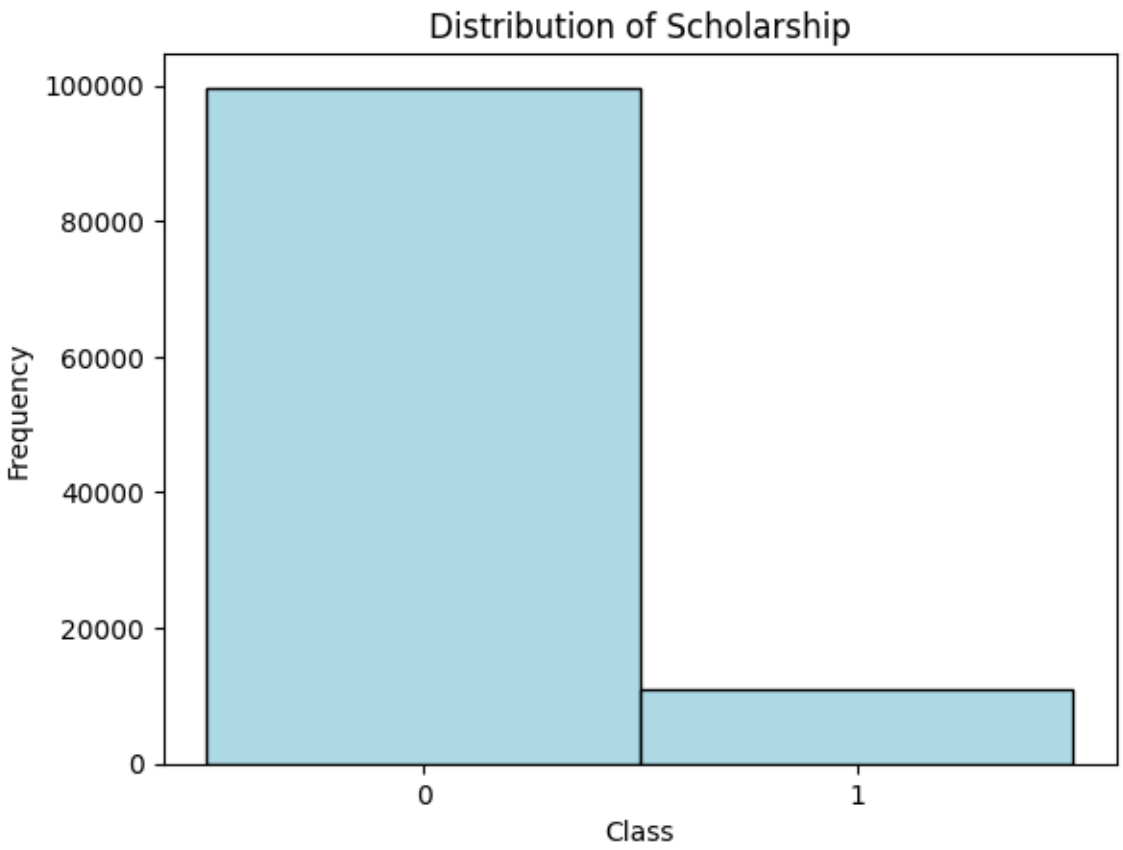
1) Data Preprocessing:
    a) Handling null values:
    In this step which is almost the first part of the whole project, we're checking for the null values before anything else. At first, it seems that there aren't any null values in the dataset, but this can be incorrect truly. Sometimes in some columns which represent two different values merged together there may be a part missing, but other not. So when we check for null values we get a kind of false negative answer for the occurrence of null value.
    In this case, we should separate these parts and store them in distinct columns and check for null values again.

That’s why I defined a function for checking null values which is called “null_search”, so I won’t have to duplicate any code to check for null values in dataset again and again.
Since we have two columns which are consist of date and time, again I defined a function for splitting a column into two new columns, one for date and one for time.
Even after all of these it seems there are no null values in the dataset, so we can move on to the next part.

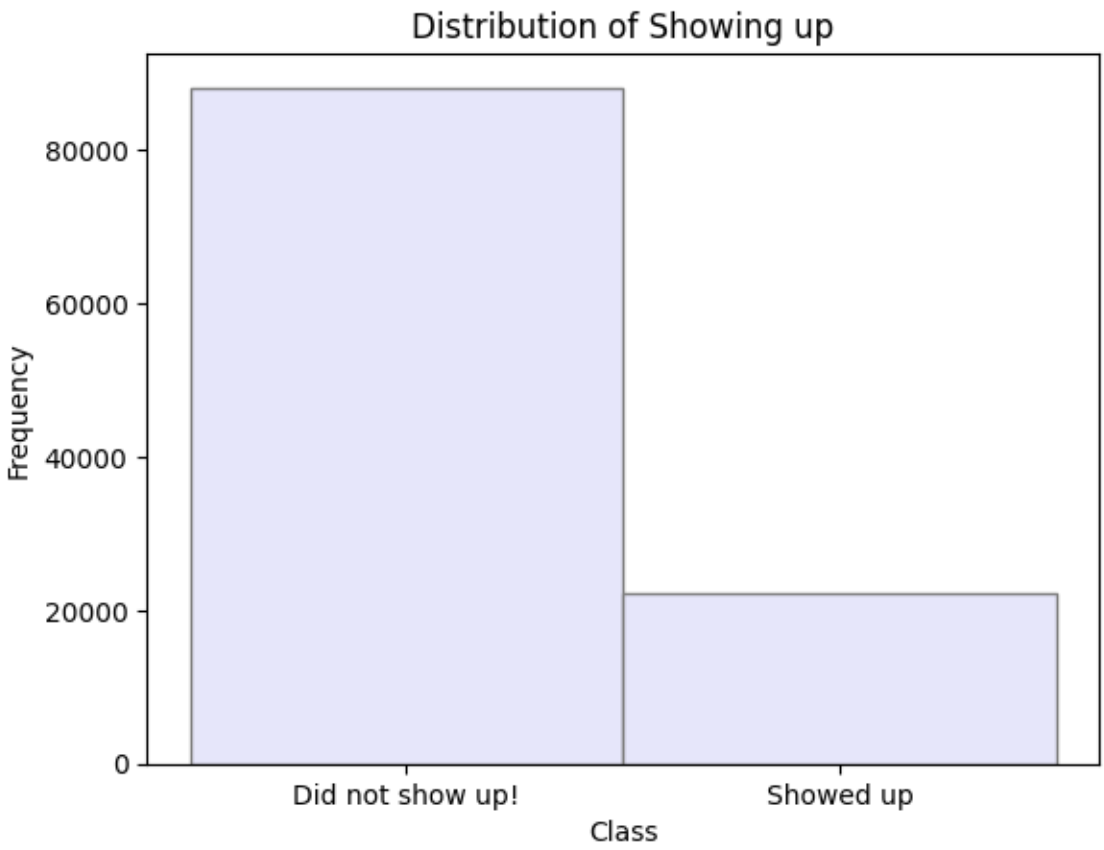
b) Imbalance Classes:

It’s typical for datasets to have imbalanced classes. This means that for a categorical feature the frequency or in other words the number of samples that belong to one of the categories is significantly more than the frequency of other classes. For instance, when 90 percent of the samples in the dataset belong to the “diabetes” category.

One of the ways to detect imbalance classes is to plot the distribution of each feature and see if frequency of different classes differs highly or not. Here are the results:


Distribution of Scholarship
100000
80000
60000
40000
20000
0
Frequency
0
1
Class


Distribution of Hipertension
Frequency
80000
60000
40000
20000
0
0
1
Class



c) Sampling:

Since our dataset is quite imbalanced for the 'No-show' classes, we must get rid of this problem otherwise it will affect our model's performance and lead to low F1-Score. A good way to handle imbalanced classes is to take a sample from data in a way that all of the samples of the minor class are included but, only some of the samples of the major class are taken. Here we take the half of the

number of samples for the major class and all of the samples for the minor one.

d) Encoding:

First of all, the ‘No-show’ column must be converted into binary form, so the MLP model can interpret it later. The same thing should be done for the ‘Gender’ column.

For ‘Neighborhood’, ‘Sch_Date’ and ‘App_Date’ we use one-hot encoding to minimize the loss of information. Using methods like label encoding may seem better due to the fewer number of added columns, but they lead to poor model’s performance.

For the scheduled date and time, it might be a good choice to combine these two features and use the new feature for the model training. The subtle point is that this combination should be sensible based on the context of data. Since the whole dataset is related to one year ‘2016’, then the year part of the date is worthless. We decided to extract the month from date and store it as a new column so it can be used later on. For the time column the important part of it seems to be the hour that was scheduled. So the hour is

extracted and stored in another column. At last, these two columns are concatenated and together they form a new feature.

2) Defining and Training the Model:

In this part we're going to define the structure of our MLP model like the number of hidden layers, the activation functions and so on.

But before starting the process I stored the final output data frame of the previous part, so I can use it later.

First, I defined a class for taking the data frame and split it into features and target columns as X and y. Then, I converted both of x and y into tensors that can be interpreted by the model. There's also a method defined for splitting the data into training and testing set by the given ratio. It's typical to use 80 percent for train and keep 20 percent of the data untouched for testing.

Next, a method is defined that uses data loader to slice our train and test data into batches with given size.

Now it's the time for defining the model's structure, I did this by using a class. There are three layers for

this model the activation function for the first two is 'ReLU' function and for the last one is sigmoid since the problem is a binary classification. Also we have to define a method for the forward pass in this class. We then build the model considering the fact that our dataset has 11 features so the number of inputs for the first layer is 11.

For the actual training process again a method is defined, having the whole code of each part as a method or class helps with the prevention of complexity of code and it makes it easier to understand and apply. In this part we must first choose our loss function and then our optimizer which are BCE and SGD in this case. Other options are also available and suitable as well, but that's what I decided to go with. Then, we iterate through the dataset and the backpropagation process is done in each iteration. The whole process is done for the set number of epochs.

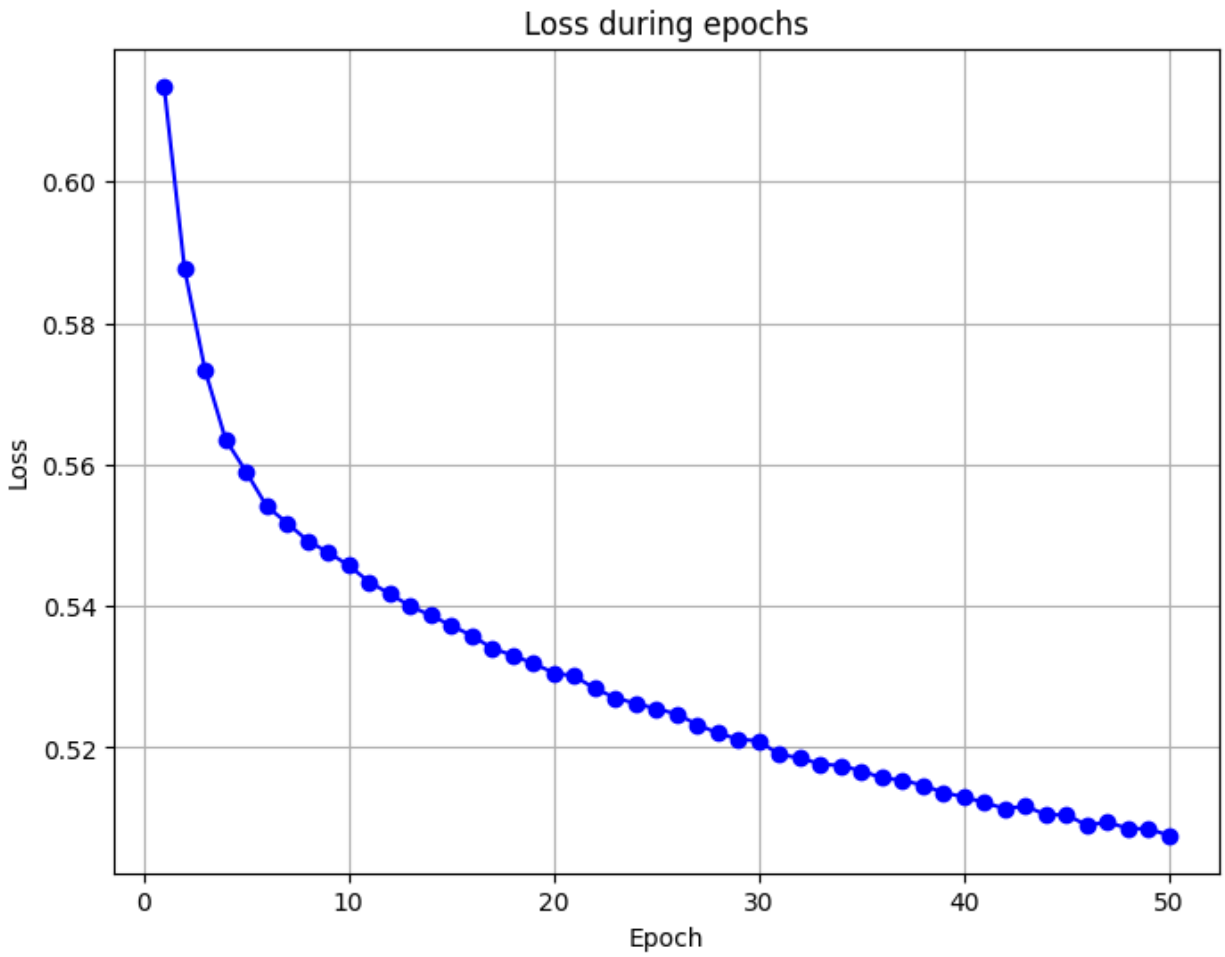
3) Evaluation:

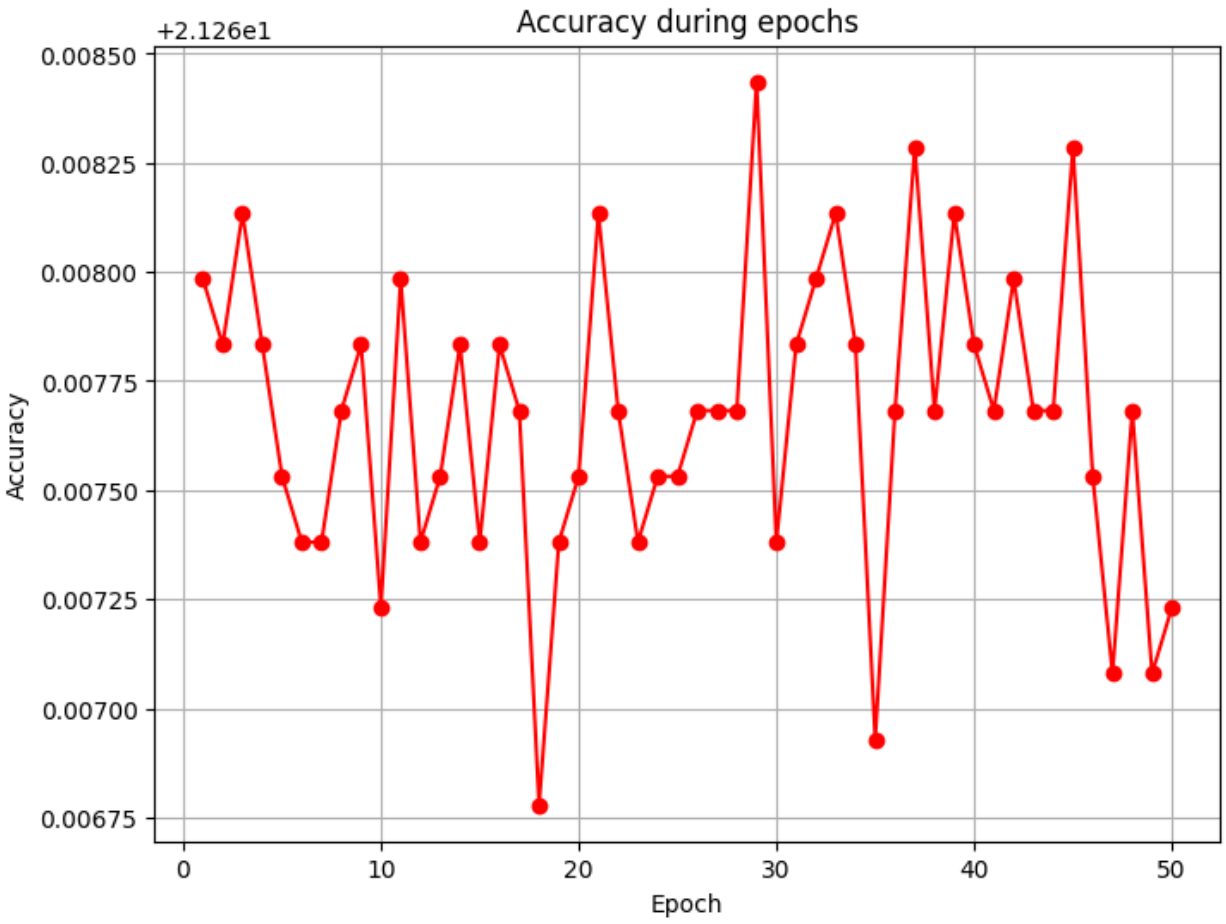
In this part we evaluate our model and see its performance on unseen data (testing data in our case). There are many evaluation metrics that we can use but, we have to select suitable metrics based

on our dataset and model. Here, we define a method for evaluation which includes accuracy, recall, precision, F1-score, True positive rate, False positive rate and others. As you can see the accuracy for our model is nearly 0.8 which is acceptable.

Evaluation Metric Type	Metric
Accuracy	0.789917
F1 Score	0.157532
Precision	0.437500
Average Precision Score	0.226856
Recall	0.096060
True Positive Rate	0.096060
False Positive Rate	0.031744
Misclassification Rate	0.210083
Sensitivity	0.096060
Specificity	0.968256

4) Visualization: At last we plot the trend line of Accuracy and Loss during epochs. As you can see the accuracy in the training process is nearly stable and the training loss decreases over epochs.


Loss during epochs
Loss
0.60
0.58
0.56
0.54
0.52
0
10
20
30
40
50
Epoch


Accuracy during epochs
+2.126e1
0.00850
0.00825
0.00800
0.00775
0.00750
0.00725
0.00700
0.00675
Accuracy
0
10
20
30
40
50
Epoch

The End