**Introduction:**

In the following lines I will explain how do I think to solve this classification problem.

**Preprocessing stage:**

I have read the train and development sets checks if there are NaN values in my own data and filling the NaN values according to the feature data type. With the string data type, I used the most frequent strategy (mode) while with floats I used the mean strategy and with the Integers I used the median to ensure replacing the NaN value with integer as this would be more reasonable. I have encoded all the strings features as well as my target with numbers so that it can be a useful input to my model. I have scaled all my features with number data type, so that the model optimizer can converge much faster.

**Development cycle:**

Trying number of models like ANN, Logistic Reg., Random Forest and SVM I have found that my model is suffering from overfitting. The training performance was 1 and the development performance does not exceed 0.55 despite tuning my parameters. I have decided to go back to the preprocessing stage applying feature selection to eliminate the dummy features that may cause high variance to my model. Using SelectKBest module with Chi2 score function as I find all my features with positive values so I see Chi2 would be suitable. Trying out different values of feature with tuning the parameters of the Random forest I have reached a desirable performance.

**Performance metrics:**

Looking up at my training targets I have noticed that the majority label is 1 but the development labels are near to each other in occurrence so I thought depending on the accuracy only could deceive us because of this I have created a performance data frame on my code that displays accuracy, Precision, Recall and Specificity. I was watching them all during my development cycle to help me reach a desirable performance.

**Trying to improve performance on my development set:**

After noticing that the majority label of my training set is 1 while the development set labels occurrence is nearly equal. I thought that can make my model bias to choose one rather than zero on testing. This leads me to think that I should make their occurrence on my training set nearly equal too to resemble the real test but this would unfortunately leads to deleting some of my samples that have the label 1. I find 600:200 training to validation is good ratio and achieves my target. I have found an improvement in all metrics except Recall it decreases only by 1%. The following table shows us the before and after eliminating some samples.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Accuracy | Precision | Recall | Specificity |
| Before | Train | 0.998649 | 0.981884 | 1 | 0.998542 |
|  | Test | 0.87 | 0.803738 | 0.945055 | 0.807339 |
| After | Train | 0.985 | 0.98913 | 0.978495 | 0.990654 |
|  | Test | 0.88 | 0.831776 | 0.936842 | 0.828571 |