The code I have shared is implementing the Naive Bayes algorithm on a pharmaceutical dataset.

The dataset is first loaded and missing values are removed using the "na.omit()" function. Then, categorical columns are converted to numeric using "ifelse()" function. The continuous variables are standardized using z-score normalization.

The Naive Bayes algorithm is implemented using the "naiveBayes()" function from the "e1071" package. The dataset is split into training and testing sets using a 70:30 ratio. The model is trained on the training set and then tested on the testing set using "predict()" function.

The confusion matrix is generated using "table()" function and then the accuracy, precision, recall, and F1-score are calculated using the formulas provided in the code.

As we can see, the accuracy of Naive Bayes algorithm dased on this dataset isn’t performs very well, this because Naive Bayes assumes that the features are conditionally independent of each other. This assumption may not hold in real-world datasets, where features may be correlated with each other. This can lead to suboptimal performance if the dependencies between features are strong.

Naive Bayes is known to perform well on large datasets, especially when the dataset is well-balanced and the features are independent or weakly dependent on each other. As the size of the dataset increases, the algorithm can learn more accurate probability estimates and better capture the underlying distribution of the data. This can lead to better classification performance and higher accuracy.

ANN：

Like the previous ones, the dataset is first loaded and missing values are removed using the "na.omit()" function. Then, categorical columns are converted to numeric using "ifelse()" function. The continuous variables are standardized using z-score normalization.

The first argument, "Survived\_1\_year~.", specifies the formula for the neural network model. In this case, "Survived\_1\_year" is the dependent variable (i.e., the variable we want to predict), and the "~." notation indicates that all other variables in the "training" data frame should be used as independent variables in the model.

The second argument, "training", specifies the data set that the neural network should be trained on.

The third argument, "hidden=5", specifies that the neural network should have one hidden layer with five nodes.

The fourth argument, "threshold=0.05", specifies the threshold value for the partial derivatives of the error function. This threshold determines the stopping criteria for the neural network training algorithm. Specifically, training stops when the partial derivatives of the error function with respect to the weights of the connections between neurons are all smaller than the threshold value.

Overall, this line of code is creating a neural network model with one hidden layer and five nodes, and training it on the "training" data set using the specified formula and threshold value.

The accuracy of ANN is 75.87%, ANN can be computationally expensive to train, especially if the network has many layers or nodes, if I add another hiden layer in this method, It will take nearly 9 minutes to run. ANNs can be prone to overfitting, where the model becomes too complex and starts to memorize the training data instead of learning generalizable patterns.

Althouth this algorithm may have some limitation,it’s still a powerful class of machine learning algorithms that can model complex relationships in the data.

Overfitting: ANNs can be prone to overfitting, where the model becomes too complex and starts to memorize the training data instead of learning generalizable patterns.