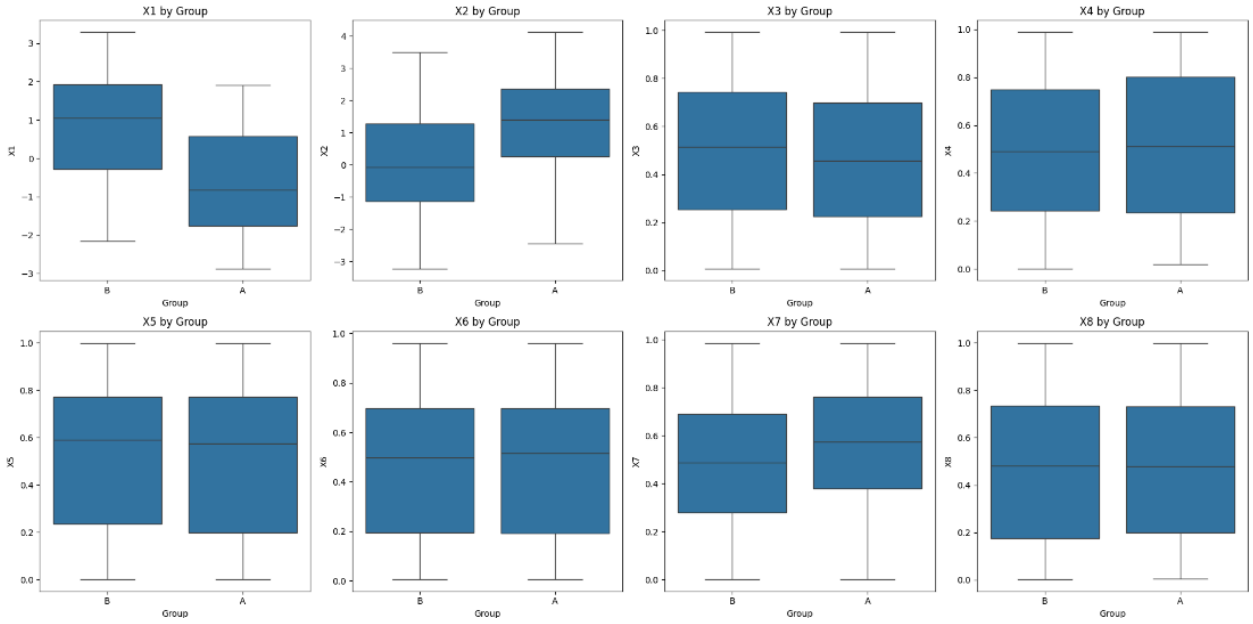
Analysis

This data has 9 columns or variables with 8 of them being arbitrary variables labeled X1 to X8 with numbers ranging from -3 to 4.5 depending on the variable. The 9th column is the group that each data point belongs to: A or B. Plotting each of the 8 arbitrary variables in a boxplot by group A or B can reveal a difference in the distribution of each group.



Some of the columns like X4, X5, and X8 have similar distributions, and might not contribute much to the groupings. However, all the other variables show different distributions from A to B with different ranges, averages, and IQRs, resulting in possible contributions. Since all columns were being used for predictor or predicted variables, no data manipulation was needed. In addition all values were not null, so no need for dropping/filtering any data.

Methods

First step in making the model was separating the data into training data and testing data using a 80-20 train-test split. Next was transforming all the data by standardizing all the data through a column transformer. We can use this column transformer for all 3 our models. The three models I will be using are Support Vector Classifier (SVC), Linear Regression (LR), and K-Nearest Neighbors (KNN).

Starting with SVC, I make the pipeline, and run a grid search to find the best hyperparameters. The 3 hyperparameters of C (margin size when classifying data), gamma (decision boundary), and kernel (conversion into higher dimensional data) are inputted into grid search to find the best combination of the 3 based on the resulting accuracy scores. Values:

```
param_grid = {'svc__C': [0.001, 0.01, 1, 5, 25, 50],
              'svc__gamma': [0.001, 0.01, 0.1, 0.5, 1, 2, 5],
              'svc__kernel': ['linear', 'rbf']}
```

This results in the best combination of values: C = 25, gamma = 0.1, and kernel = rbf. “Rbf” is short for Radial Basis function which maps data into an infinite dimensional space for better

analysis. Using the best combination of hyperparameters from above, the model was used to predict the test data.

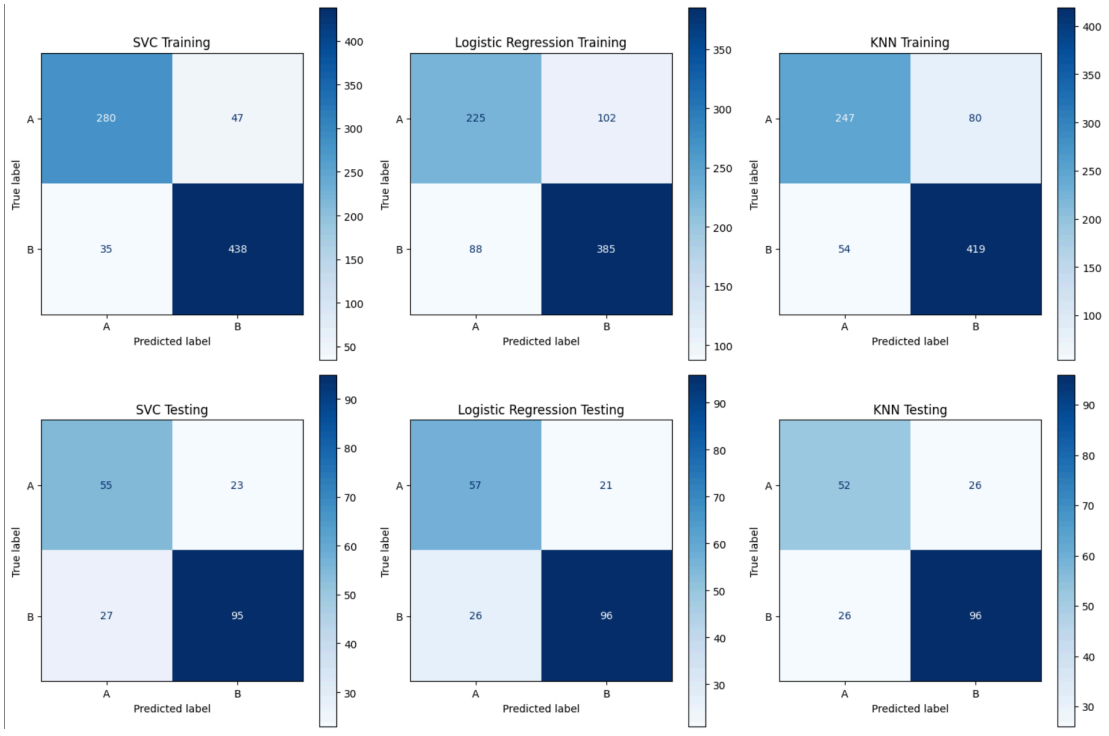
Linear Regression had a much easier process as there were no hyperparameters. A pipeline was made, fit to the training data, and then used to predict the test data. KNN follows the same process as SVC but it uses the hyperparameter of k values which is how many neighbors the model takes into account to classify a datapoint. Values of 1, 2, 3, 5, 10, 15, and 20 were tested. Based on the highest accuracy score, 5 was the best k value. Using 5 as the k value, the test data was predicted.

Results

The following accuracy scores and ROC/AUCs were computed for each model:

Results	SVC	LR	KNN
Train Acc	0.898	0.763	0.833
Train ROC/AUC	0.968	0.841	0.909
Test Acc	0.75	0.765	0.74
Test ROC/AUC	0.818	0.855	0.797

These results show that all 3 models mostly detect the trends in the data, and can mostly identify which group a data point belongs to from the high metrics. Linear Regression is the best model based on the highest accuracy and ROC/AUC scores on test data. Even though SVC has the highest training metrics, it loses some accuracy when inputting unseen data (test data). This can mean that the model tends to be overfit. The same trend can be seen with the KNN model as well. On the other hand, the linear regression model actually improves metrics slightly on test data. The following confusion matrices were produced:



The same trends can be visualized with the correctly labeled groups being on the blue/dark blue diagonal of each matrix. Based on the limited data we have, I would recommend using the linear regression model as it's the most accurate with almost perfect fitting between training and testing data. Based on the data we got, it would do the best against unseen data compared to the other 2 models. However, the higher accuracy trends seen in SVM's training data shows signs of potentially being a better model if more data is introduced. The overfitting seen in the SVM and KNN model could be a lack of quantity of data, but until we get more data, it is indeterminate.

Reflection:
Aside from the usual syntax errors that were fixed in due time, something that I did find that I could improve on, is the naming conventions of variables. Since we were using 3 different models in the same file, and a lot of them use similar code, I found myself going back and renaming every variable for each specific model. It would have saved a lot of time if I just initially named every variable with the model abbreviation at the end to help with readability, and making sure correct variables were used. Otherwise, I learned how important data can be to models, as 2 of the models were overfit, which were likely from the amount of data. The simple model of linear regression happened to do the best on the little test data, while the more complex models, like SVM and KNN, did really well on lots of training data but worse on little testing data (overfit). Lack of data might not be the whole reason, but considering the lack of noise (normal distributions of box plots / large IQRs), and not that complex of models, I suspect it might be.