Homework 1

Spencer Au

Analysis

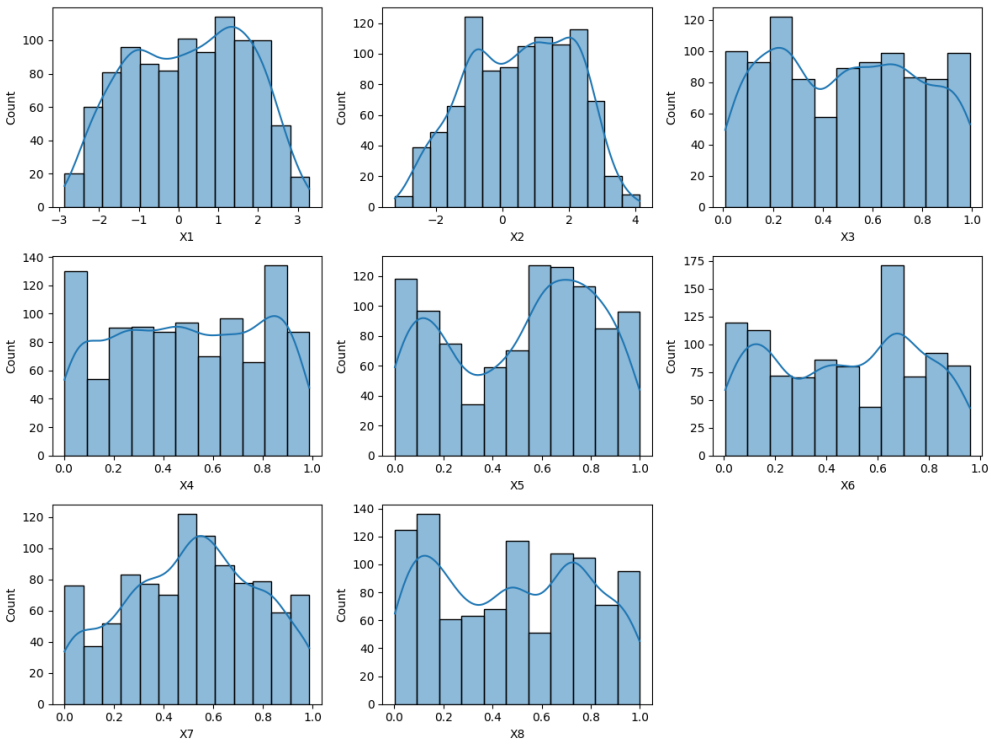

Figure 1: Distribution of Each Variable

The data is made of 8 variables, numbered X1 to X8, and the predictor variable is Group. The predicted variable is a binary variable, with A and B as the two possible values. The data is relatively clean, with no missing values. Overall it seems like X1, X2, and X7 have a normal distribution, while X3, X4, X5, X6, and X8 have a bimodal distribution.

Methods

We used three models: Support Vector Machine, Logistic Regression, and K-Nearest Neighbors. All three models were trained and tested on a 80/20 split of the data, which means that 80% of the data was used to train the model, and 20% was used to test the model.

Support Vector Machine

For the SVM, we used GridSearchCV to choose the kernel (from linear and rbf), C (from [0.001, 0.01, 1, 5, 25, 50]), and gamma (from [0.001, 0.01, 0.1, 0.5, 1, 2, 5]). The "kernel" is a setting that changes how the SVM looks at your data. It's like choosing between an Android or an iOS operating system; both are good, but they work in different ways. In this case, we're choosing between "linear" and "rbf" (which stands for Radial Basis Function). A linear kernel looks at the data in a straight-line fashion, while rbf can handle more complex patterns.

The "C" value is like setting a budget for your phone. It tells the SVM how much you care about making mistakes. If you set it too low, it's like being too thrifty and ending up with a phone that doesn't work well. If you set it too high, it's like overspending on features you don't need. The value of "C" helps the SVM balance getting the classification right with keeping the solution as simple as possible.

The "gamma" value is a bit like the zoom level on your phone's camera. It controls how closely the SVM looks at the data. A low gamma means it looks from far away, making broad generalizations (like a wide-angle shot), while a high gamma means it looks very closely, paying attention to all the tiny details (like a close-up).

Now, to find the best combination of these settings (kernel, C, gamma), we use something called GridSearchCV. Imagine it as a super organized friend who makes a comprehensive list of smartphones with all possible combinations of features and prices, then methodically goes through each option to find the best one. GridSearchCV does that for us with SVM settings; it tries out all combinations of kernel, C, and gamma values we give it, then tells us which one works best for our data.

The Hyperparameters we chose were: kernel = "rbf", C = 50, gamma = 0.001

Logistic Regression

For the Logistic Regression, we did not use GridSearchCV. Logistic Regression is a simple model that doesn't have many settings to tune. It's like a basic phone that doesn't have many features to customize. We used the default settings for the Logistic Regression model.

The Hyperparamters we chose were: None/Default

K-Nearest Neighbors

For the K-Nearest Neighbors, we used GridSearchCV to choose the number of neighbors (from [1,2,3,4,5,6,7,8,9,10]). To explain it, its like trying to sort fruit by color. If you have a lot of fruit, you might want to look at the color of the 3 closest fruits to decide what color to call the fruit. If you only look at the closest fruit, you might make a mistake if that fruit is a weird color. If you look at too many fruits, you might make a mistake if the fruit is a normal color. The number of neighbors helps the KNN model decide how many fruits to look at to make a decision.

The Hyperparameters we chose were: n_neighbors = 9

Results

Metric	SVM - Train	SVM - Test	Log Reg - Train	Log Reg - Test	KNN - Train	KNN - Test
Accuracy	0.77625	0.67	0.7825	0.685	0.8125	0.68
ROC	0.86321	0.7691	0.86287	0.76544	0.8998	0.72871

Overall, it seems that all three models performed better on the training data than the testing data. This is expected, as the models are trained to fit the training data, so they should perform better on the data they were trained on. However, both the SVM and Logistic Regression models performed similarly on the testing data. Its kind of like when a basketball team does really well in training, but then does not perform as well when its an actual game. Accuracy is a measure of how many predictions the model got right, and AUC/ROC is a measure of how well the model can distinguish between the two groups.

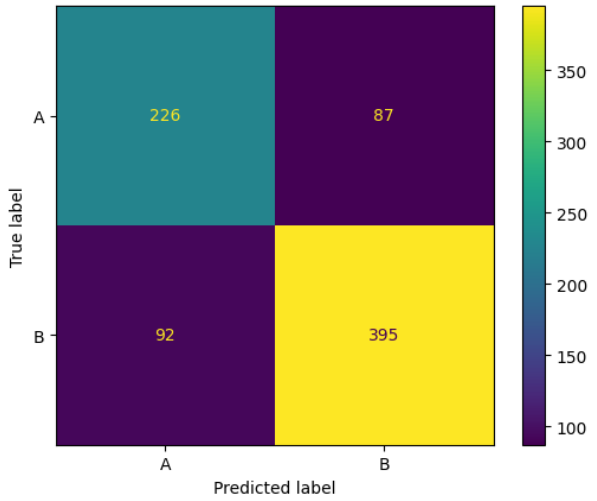

Figure 2: Confusion Matrix for SVM

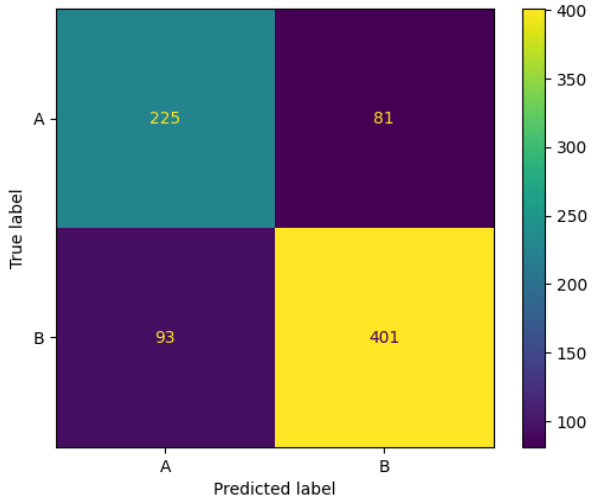

Figure 3: Confusion Matrix for LR

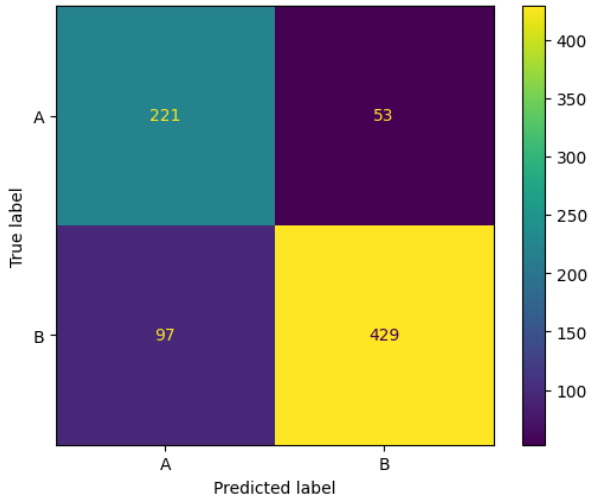

Figure 4: Confusion Matrix for KNN

Looking at the respective confusion matrices, we can see that the SVM and LR models performed similarly, with the KNN model performing slightly better, especially in terms of predicting Group B when the actual group was Group B, and not predicting Group B when the actual group was Group A.

Overall, I would suggest using KNN model in production, as the three models have similar performance, but the KNN model performed noticeably better on Group B predictions.

Reflection

In terms of what I learned, it doesn't seem like the SVM model performed as well as I had expected. I had thought that the SVM model would perform better than the other two models, but it seems like the KNN model performed the best. I was surprised by how well the Logistic Regression model performed, especially given that it was the sole model not to use GridSearchCV.