# Narrative

SVM is a very efficient and accurate algorithm that maps data so it can be analyzed. The algorithm first plots the data into an N dimensional space and then attempts to separate the categories. This is done by finding the hyper plane that creates the largest margin on either side. This margin is the distance from the hyperplane to the closest node on either side of the separation. Once the data is separated into the categories, it then shifts the data so that the separating hyperplane is in one of three shapes. With a linear kernel the separating hyperplane is a straight line or plane that separates the two different sections and is good when the data is linear and easily predicted via the predictors. Sometimes the data is nonlinear which requires a different shape for the separating line, into one of two options. The first is a polynomial kernel that is used when, for instance, one section that is plotted is generalized into the center of the graph and the other section is on either side of it, making it so that a curved hyperplane fits better for the data rather than a linear one. The last possibility is one where one section of the data is surrounded by the other section of data, meaning that a circular or radial hyperplane best fits for the data. With the three different types of SVM that can be used, there are lots of applications for the algorithm in all sorts of different data sets. SVM works best when the there is a clear separation between the two different categories and when there are more predictors being used. The higher dimensionality of the graph means that there are more ways for a hyperplane to be drawn between the two categories. However, the downsides of SVM are not something that can be overlooked completely. SVM is very slow with large data sets or with data sets that are not as correlated and overlap too much, making it run very slow and be less accurate than it should be.

Random forest works by creating a multitude of different trees that each predict the target based on the predictors, and the class with the most guesses is the output. This works well because with all of the different trees, each has their own slight differences that can help catch things like edge cases or stop bias by spreading out the potentially biased data to the point where it has less of an effect. Group think

algorithms like this are very potent because they will always be more accurate than their counterparts as long as each tree is not correlated to each other. XGBoost algorithm is similar, but instead of using just the data given, it also uses gradient boosting to try and get better results. This gradient boost is helpful because it ensures that the model is better created by iterating through and creating new models by taking the model before it and basing the new model off of it. Boosting is very similar to XGBoost, but without the tree aspect. It creates model after model and every new model tries to be better than the model that came before it. The XGBoost algorithm is the combination of both the Random Forest and the Boost algorithm and is better than each of its parts individually. The combination of the two ensures that the XGBoost algorithm is a very powerful one with broad applications that are used throughout the industry. They, however, aren't perfect. Random forest begins to fail if the data overlaps too much and each individual tree isn't separated from each other, meaning that the entire system begins to collapse and results in subpar predictions. The basic Boost algorithm suffers from an inability to handle outliers, with each model being created attempts to fix the errors of the one before it, the model could end up overfitting to outliers making the predictions inaccurate. XGBoost has the weakness of both of the others, meaning that while it is very powerful when used on properly structured data, it can also fail easily when the data is subpar.