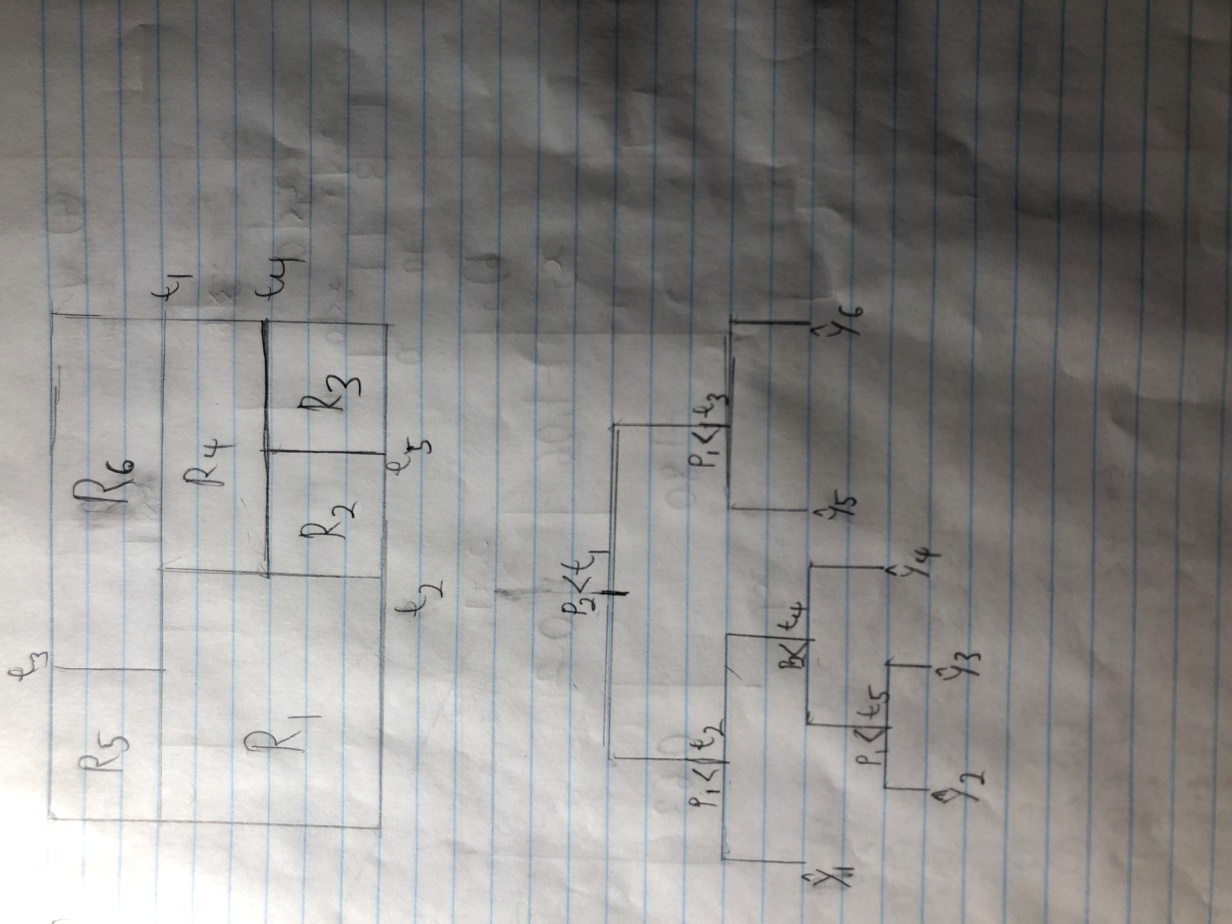
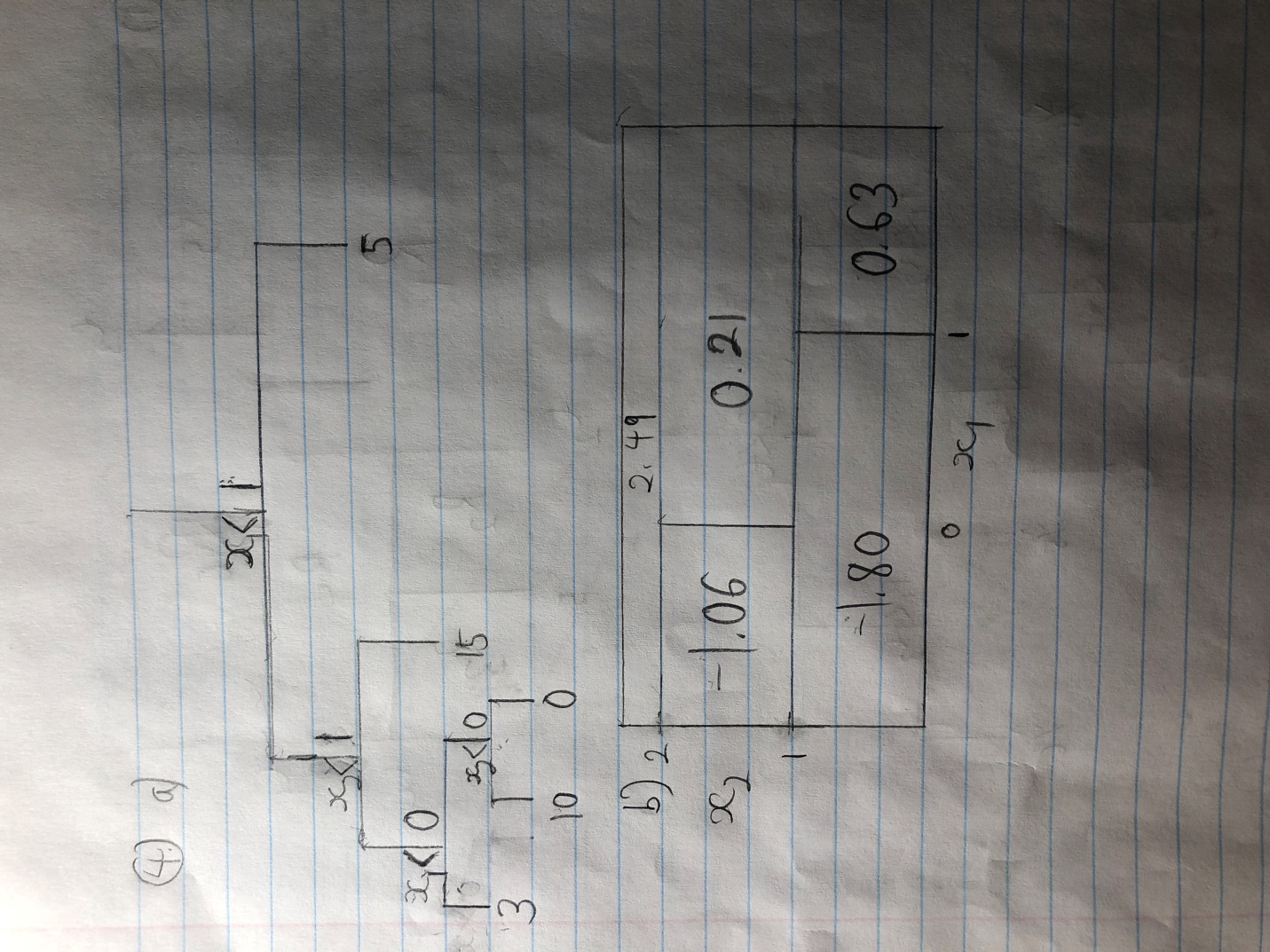
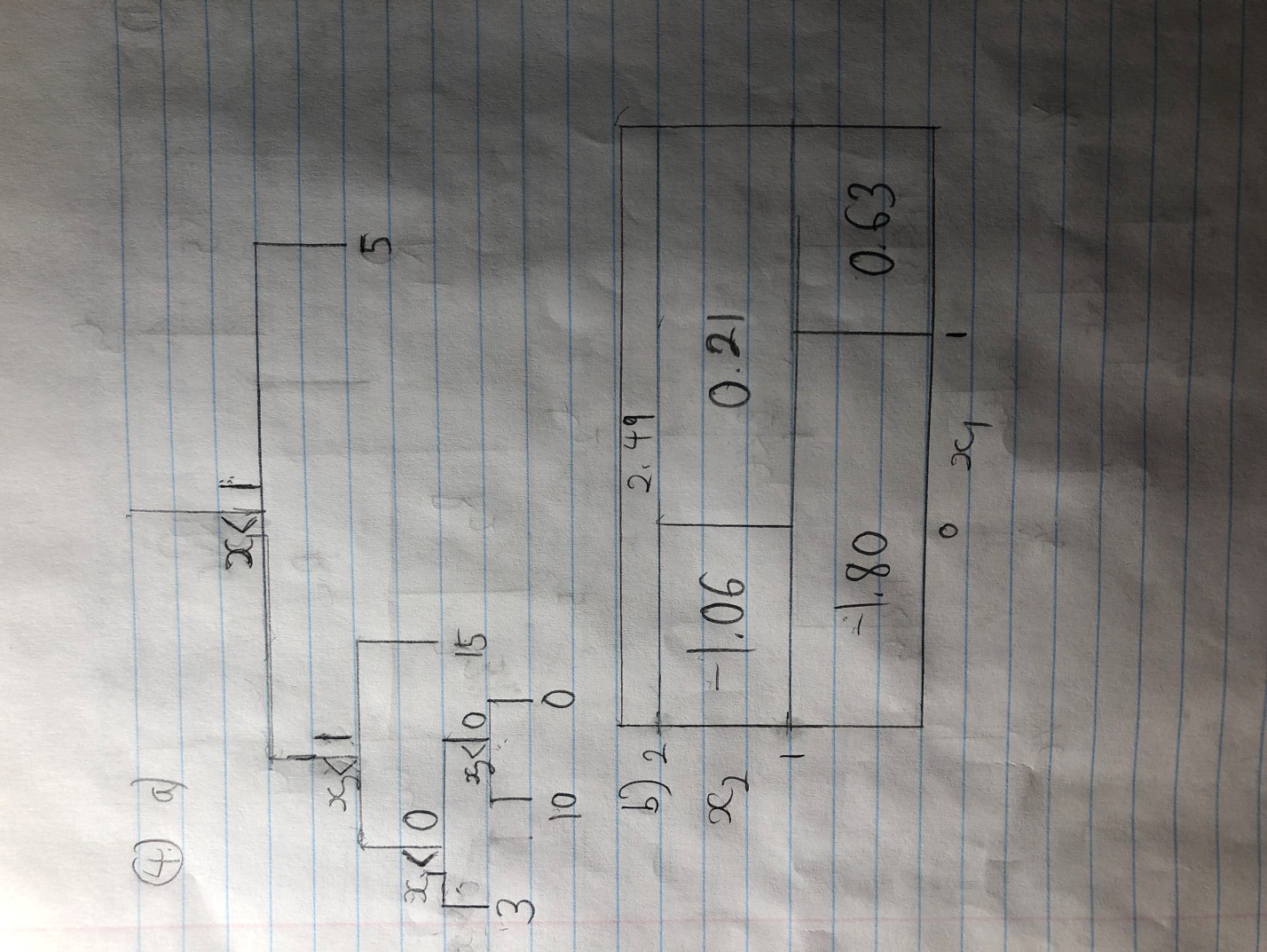
1. Trees



1. In a boosted tree model with the number of splits *d*=1, or using only stumps, then the resulting fit is an additive model due to the fact that, in decision trees, every split consists only of a single variable. In this case, every constituent boosted tree in the boosted forest is a function of a single predictor in the dataset, leading to a model of the form y=f1(x1)+f2(x2)+…+fn(xn), identical to the form of a standard additive model.
2. In R script
3. Decision Tree Sketching
   1. Left-hand side of



* 1. Right-hand side of



1. In the context given by this problem, then the two different tree classification calculations would return as follows:
   1. If we take a majority vote, 6 of the 10 bootstrapping estimates have values for P(Class is Red|X) which are greater than 0.5 or 50%. Therefore, the majority vote would return that this observation is indeed **red**.
   2. If we take the mean average of the 10 bootstrapping estimates, we would get (0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75)/10=4.5/10=0.45, which is lower than 0.5 or 50%. Therefore, the average probability method would return that this observation is not red, and is in fact **green**.
2. The algorithm for building a regression tree begins with the use of recursive binary splitting to grow a very large tree using the training data. This entails going through all of the predictors and all possible values to implement a partition of the predictor space into two distinct regions, or cutpoints, and select the predictor and cutpoint which lead to the greatest reduction in RSS. This step continues until each of the terminal nodes, or leaf nodes, has fewer than a preset minimum number of observations. Cost complexity pruning to the resulting tree is then performed, which involves using a non-negative tuning parameter *a* to index and consider only a specific sequence of trees. The tuning parameter *a* acts similarly to how lambda acts in the lasso equation, which penalizes the model and increases its lasso error for more complex models, or models which contain more variables. A similar concept is applied here, where the tree’s cost complexity value penalizes it for having more terminal nodes, thus making its error also increase with the number of terminal nodes it possesses. This means that the number of terminal nodes *T* that minimizes its error is often less than the number of terminal nodes in the original large tree. In turn, it can be deduced that a smaller subtree will minimize this quantity rather than a larger one, allowing the algorithm to cut off the branches from the original tree that do not exist in the optimized subtree. Then, k-fold cross-validation is used to select the optimal value of *a* in order to make the tree creation algorithm as effective as possible. First, we divide the training data into *K* distinct folds; then we repeat the first two steps of growing the tree and then pruning it on all but the *k*th fold of the data. This allows us to evaluate the MSE (with cost complexity) of the trained model’s predictions on the observations in the *k*th fold, which is a function of *a*. We repeat this sub-algorithm for all *k* from 1 to *K*. Finally, we average the prediction results for each value of *a*, and select the value of *a* that minimizes the average error. The best subtree, specifically the one that minimizes the cost complexity function with the optimal, aforementioned value of *a*, is selected and returned as our final regression tree.