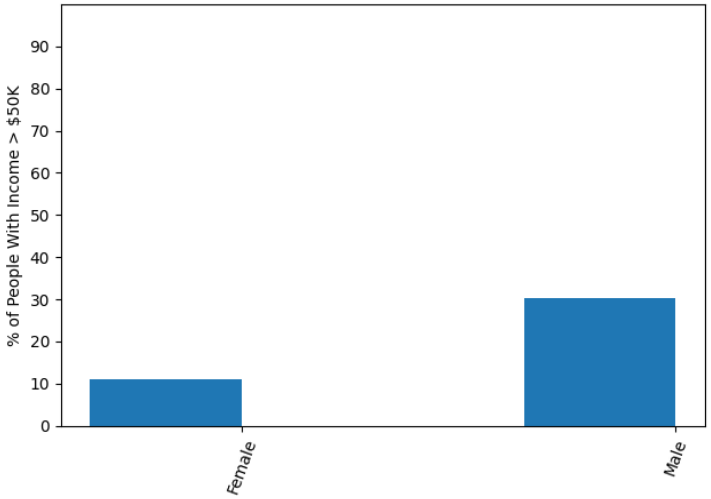
Bhavya Gupta, Ananya Kashyap, Viraj Vijaywargiya
34333649 14686673 25469141

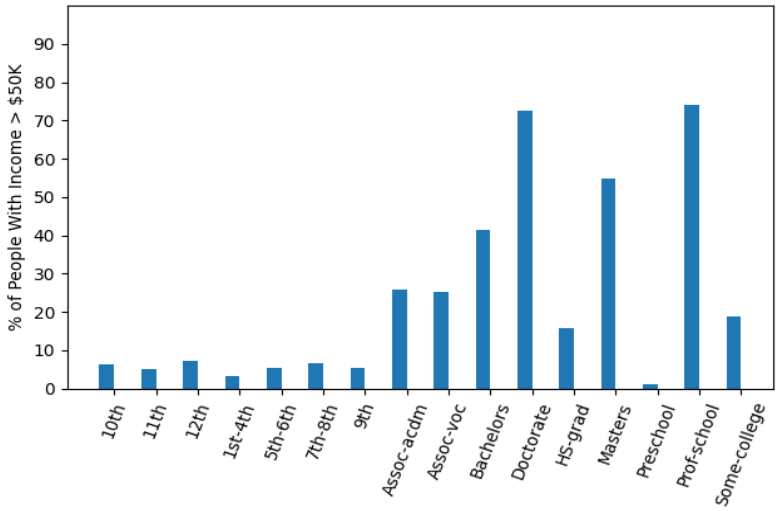
Comparative Analysis: Gradient Boosting & AdaBoost

Dataset: Adult from the UCI Machine Learning Repository
Classification: Binary classification whether income exceeds $50K/yr based on 1994 census data (Census Income).
Learners: AdaBoost & GradientBoost
Purpose: Explore, between Gradient Boosting and AdaBoosting, which model classifies the data better.

Preliminary Findings & Data Exploration

We initially ran tests to check the relationship between each feature by itself to check its correlation with the output of income. Some finds were expected, such as the graphs below (sex, level of education). These are also examples of features with high influence on the target values, as evident by the significantly varying bar heights.



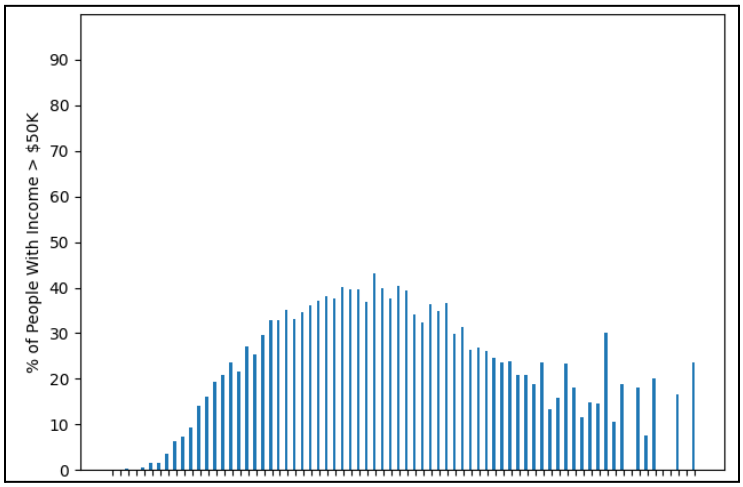


The first graph depicts that ~10% of female workers make >50k USD yearly, while ~30% of males do. Given that this data is also pulled from a 1994 Census, the results make sense. At that time, there were far more male breadwinners as compared to female breadwinners, and as such males were likely to make more income to support their families. Additionally, women tended to have lower paying jobs for a variety of reasons, including discrimination, average skill set, and an undervalue of labor.

The second graph clearly shows that income rises with education level. With more education, an individual has more available roles that often require more knowledge in the field. As such, the jobs in question pay more to the fewer professionals who have taken the time and spent the money to get educated in a niche.

Other finds were a little more unintuitive. For example, the graph below uses age as a feature shows more of a binomial distribution where median ages have the highest percentage of income over the

threshold. This may be because older people have a decline in physical/mental ability, or lack the need to generate as much income as before. They may need a lower income to sustain themselves now, or might just work in an area of passion that pays lower.



Lastly, let's look at some issues we encountered. There were many data points with missing values, especially in the workclass, occupation, and native country features. As such, we opted to remove incomplete datapoints since this resulted in better accuracy for our model. Additionally, some data cleanup was required in terms of data types (some values were strings while others were ints/floats) as well as inconsistencies within a type of data ('50K' vs '50K.'). Preprocessing resulted in a significant boost in how our models performed overall.

Having thoroughly explored our dataset, we now transition to comparing two widely recognized ensemble models, **Gradient Boost and AdaBoost**. This comparison will involve carefully training and evaluating each model, allowing us to discern their strengths and weaknesses in the context of our specific data.

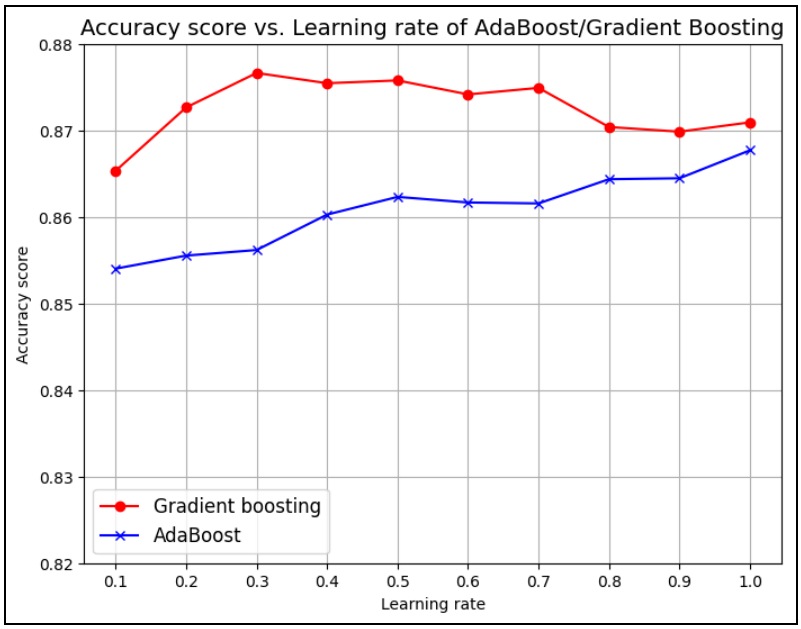
Learning and Evaluating Gradient Boost and AdaBoost Models

Gradient Boost is an ensemble learning method that uses several weak learners to create a stronger model by sequentially adding new models to correct mistakes made by previous ones. AdaBoost on the other hand is another ensemble-based method that instead weighs misclassified data more than correctly classified data. The model aims to fix these errors more efficiently in future iterations, and the weighted combination of these learners provides us with the final model.

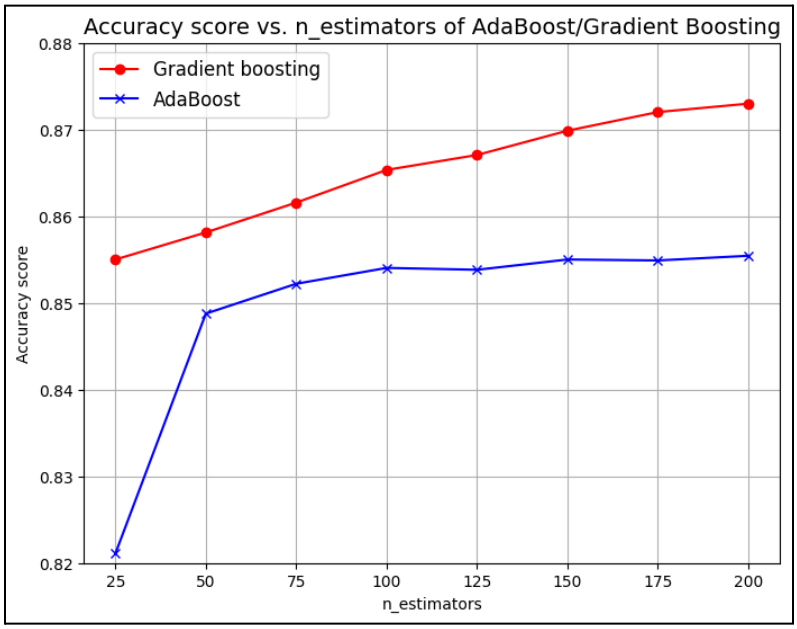
To compare the models, we visualize their accuracy scores against the two main parameters, *learning_rate* and *n_estimators*. Focusing on these parameters provides a more straightforward, interpretable, and computationally efficient way to analyze and compare models, especially when the goal is to understand the fundamental differences in their learning behaviors. For our tests, here are the parameter values/ranges used:

- *random_state:* 42
- *max_depth*: default
- *n_estimators* (Constant for *learning_rate* graph): 100
- *n_estimators* (*n_estimators* graph): [25, 200], increments of 25
- *learning_rate* (*learning_rate* graph): [0.1, 1.0], increments of 0.1
- *learning_rate*(Constant for *n_estimators* graph): 0.1

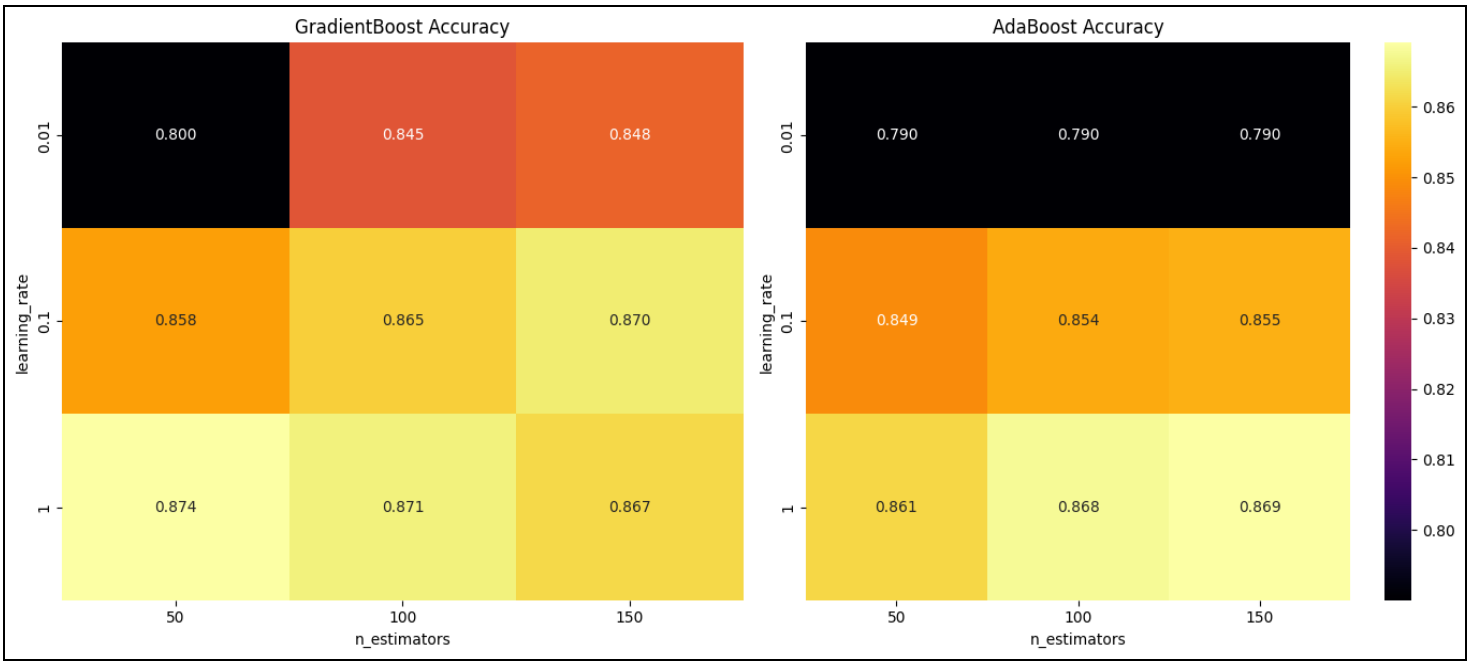
Graphing the models over various learning rates reveals key insights into their performance dynamics. The learning rate indicates the contribution of each estimator to the final classifier. Initially, between the learning rates of 0.1 and 0.3, both models improve in accuracy. Notably, GradientBoost reaches its peak accuracy near 0.3, after which its accuracy diminishes with further increases in the learning rate. In contrast, AdaBoost demonstrates a steady improvement in accuracy indicating a benefit from a more aggressive learning approach. When the learning rate approaches 1.0, the accuracy trajectories of the two models converge, indicating a similar sensitivity to updates at this point. Across the board, GradientBoost maintains a performance lead, suggesting it may be the superior option in environments where precise learning rate optimization is unfeasible.



The graph to the right presents a comparative analysis of accuracy scores against the number of estimators for both models. GradientBoost had a directly proportional relationship with this parameter, and seemed to rise consistently. On the other hand, AdaBoost showed massive improvement during the jump from 25 to 50 estimators, but then struggled to improve past that 100 mark–*n_estimators* helps AdaBoost as well, but plateaus after a certain point. Beyond this threshold, the model did not improve its accuracy rate by a meaningful amount. This analysis highlights the importance of identifying optimal model complexity to maximize predictive performance without incurring unnecessary computational costs.



There is a trade-off between the n_estimators and learning rate – if you have a lower learning rate, you could have more estimators since each individual estimator is contributing less, but a higher learning rate would need fewer estimators since each individual estimator is contributing more. If both are high, the performance is high, but the computation cost is also high. We investigated this tradeoff and graphed the results below to determine the optimal tuning of parameters for both models.



The heatmaps above show the accuracy scores for GradientBoost and AdaBoost models with varying combinations of *n_estimators* and *learning_rate*. Each cell in the heatmap corresponds to a combination of *n_estimators* on the x-axis and *learning_rate* on the y-axis, with the color intensity and the value indicating the accuracy score. We can observe that for a higher *learning_rate* (1), both models perform differently as the *n_estimators* increase. The accuracy for GradientBoost decreases as *n_estimators* increases, whereas the accuracy for AdaBoost increases as *n_estimators* increases. This shows that there is a positive correlation between the two parameters for AdaBoost to achieve a better model, and a negative correlation between the two parameters for GradientBoost to achieve a better model, likely due to overfitting. Needing less estimators reduces the computational cost, so Gradient Boosting is achieving a higher accuracy score while minimizing the computational cost.

Conclusion

Based on our investigation into the performance of Gradient Boosting and AdaBoost, we have determined that GradientBoost performs, on average, better than AdaBoost on this dataset. In the case of using default parameters and varying the learning rate, Gradient Boost consistently had a higher accuracy score than AdaBoost. In the other case of using default parameters and varying only the number estimators, GradientBoost continued to outperform AdaBoost. Even while comparing the heatmaps, GradientBoost scored higher than AdaBoost with their respective optimal models. We have determined that AdaBoost may be suited for simpler models because, unlike gradient boosting, it is a simple algorithm and may not necessarily perform well on datasets with many features. On the other hand,

Gradient Boosting, having more flexibility, is better suited for more complex models and datasets with many features, making it the better fit for this data.

Contributions

Bhavya Gupta: Background, Preliminary Findings & Data Exploration + code for for data exploration
Ananya Kashyap: AdaBoost Exploration + code for AdaBoost Classifier training/graphing
Viraj Vijaywargiya: GradientBoost Exploration + code for GradientBoost Classifier training/graphing
Everyone: general discussion/ideas/approaches, writing the report, comparison of models and their parameters