Supervised Learning

Ian Dover

September 25, 2023

1 Datasets

1.1 Auction Dataset

Auction verification is a complex topic which has led to billions in lost revenue. Detecting undesirable auction outcomes is of upmost importance when it comes to auctioning policies. My selected dataset comes from the "UC Irvine Machine Learning Repository" (DOI: 10.24432/C52K6N). The auction dataset can be found here: https://archive.ics.uci.edu/dataset/713/. This dataset is derived by looking at bid history for a given auction, and then evaluating whether the final outcome of the auction was desirable or not. The duplicates rows in this final dataset were then removed to arrive at a final 2043 rows. This problem can be approached as either a classification or a regression problem, but for the purpose of this project, only the classification target was considered.
The features of this dataset are as follows:

- process.b1.capacity - Integer - Bidder 1: Max number of products to win
- process.b2.capacity - Integer - Bidder 2: Max number of products to win
- process.b3.capacity - Integer - Bidder 3: Max number of products to win
- process.b4.capacity - Integer - Bidder 4: Max number of products to win
- process.price - Integer - Currently verified price
- process.product - Integer - Currently verified product
- process.winner - Integer - Bidder verified as current winner

Below is a linear (pearson) correlation heatmap of the 7 features:

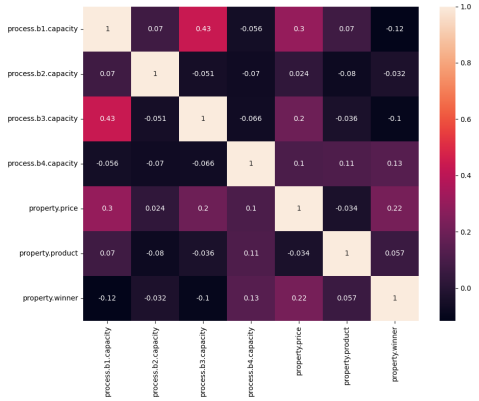

Figure 1: Auction Linear Correlation (Pearson) Heatmap

It can be seen that these 7 features are not linearly correlated, which makes sense, because individual bidder capacities should not be linearly correlated to one another when randomly assorted. Additionally, bidder capacity is unrelated to the details of a particular product in an auction.
Additionally, we take a look at the bidder 1 capacity feature with the property features as a series of bi-variate scatter plots:

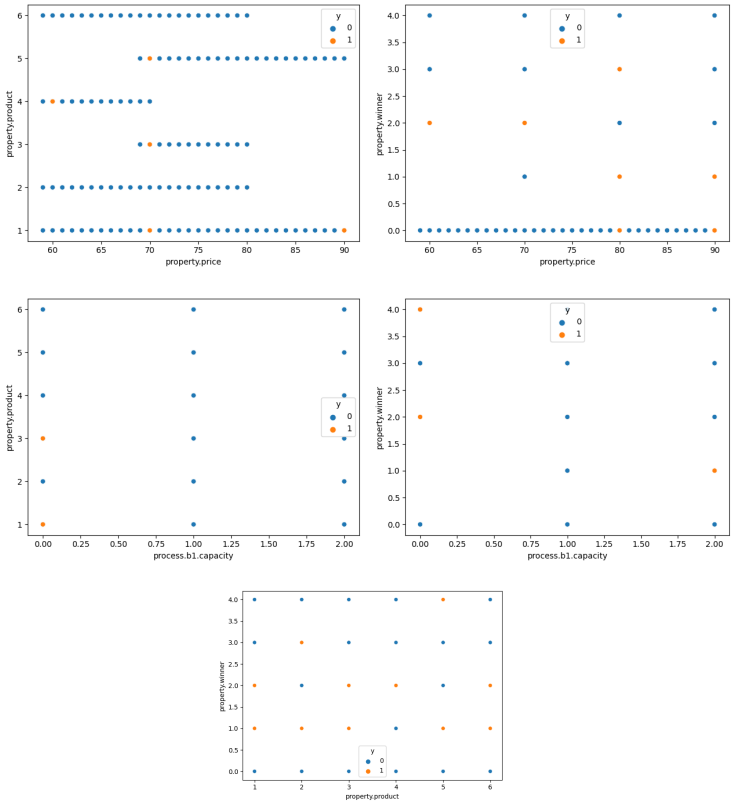

Figure 2: Scatter Plot of Top 4 Features with Hue Representing Class

As can be seen from the above scatter plots, the feature space is mostly discrete. Impossible auction values seen to be more frequent at smaller values of "property.winner", suggesting that if bidder 1 and 2 are leading the bid, then the auction outcome is impossible. This makes logical sense, because later bidders (3 and 4) should theoretically have the highest bid values. Additionally, bidder 1 capacity values of 0 tend to correlate with impossible outcomes.

1.2 Student Dropout Dataset

Predicting student dropout is essential to the enactment of mitigation efforts to improve retention rates. If we can predict a student dropout ahead of time, an intervention can be made to increase their likelihood of succeeding. This dataset contains information known on a student when they enroll, such

as: past academic performance, anticipated academic path, demographics, and socio-enconomic factors.
Each row in this dataset represents an individual student and their eventual outcome: "graduated", "dropout", or "enrolled". For the purpose of this project, the targets were converted to 0, 1, and 2, respectively.

As determine by the feature importance measure of our best performing model, XGBoost, the below features were found to be most relevant to predicting student academic success:

- Curricular units 2nd sem (approved) - number of curricular units approved in the 2nd semester
- Curricular units 1st sem (approved) - number of curricular units approved in the 1st semester
- Curricular units 2nd sem (grade) - grade of the second semester (ranges from 0 to 20)
- Curricular units 2nd sem (evaluations) - number of evaluations to the curricular units in the second semesters
- Course - courses taken
- Tuition fees up to date - flag representing financial status
- Age at enrollment - age of the student at enrollment

Below is the linear (pearson) correlation heatmap of the top features:

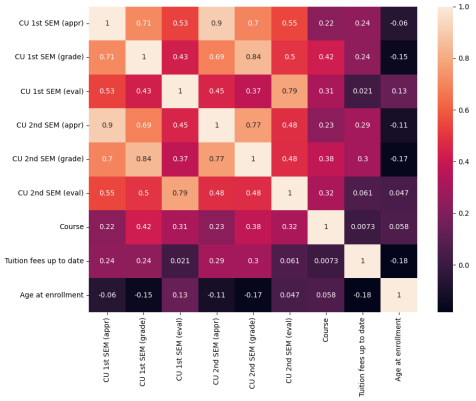

Figure 3: Dropout Linear Correlation (Pearson) Heatmap

As can be seen by the above correlation heatmap, the second semester features heavily correlate with the first semester features; however, because the 1st semester features were pushed down in feature importance in the XGBoost feature importance output, it could be said that the 2nd semester results are likely more predictive of academic success. This makes sense from a logical perspective, as student would have an additional semester to acclimate to the school environment. Moreover, the feature "Course", "Tuition fees up to date", and "Age at enrollment" seemed to be relatively uncorrelated with any of the other important features in the dataset.

Here we show the bi-variate feature distributions as scatter plots with multi-class color-coding:

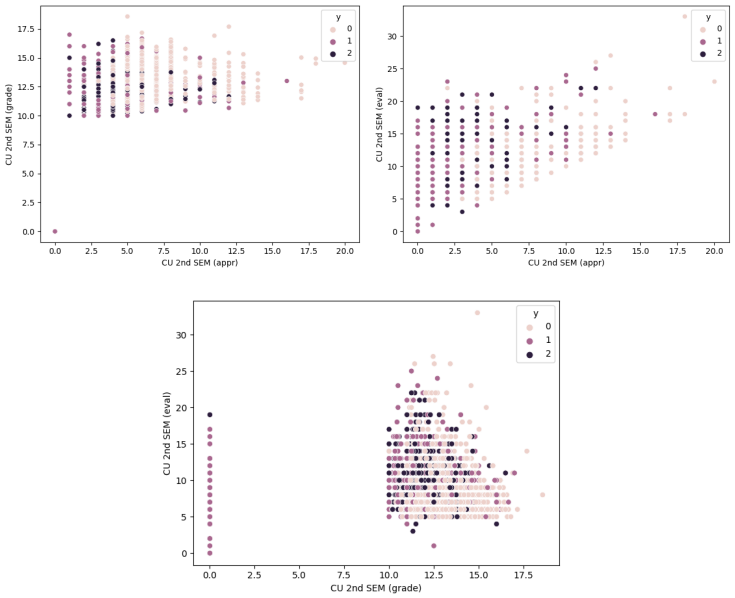

Figure 4: Scatter Plots of Top 3 Features with Hue Representing Class

Based on the above scatter plot, graduation can be predicted by a high 2nd semester grade and a low second semester evaluation. This makes sense, because a high grade is indicative of future performance; additionally, a 2nd semester grade of 0.0 almost always leads to a dropout. This trend can also be seen with 2nd semester approvals; however, this analysis may be causal, because those students with high 1st semester grades and low evaluations would have a higher approval count.

2 Decision Tree

A decision tree model was fine-tune on the auction dataset by experimenting with the "CCP_alpha" parameter, which ranges between 0.0 and 0.05. This parameter helps control the model's complexity. As I increase "CCP_alpha", I notice that both the training and testing AUC (Area Under the Curve) decrease and become pretty similar. However, the accuracy of the model on both the training and testing data does not drop much and stays nearly the same. This might be happening because, with a higher "CCP_alpha", the model becomes more generalized and less sensitive to the training data, which can decrease variance. The smaller drop in accuracy compared to AUC could be because the model remains pretty good at making correct predictions, possibly due to prevalent classes or simpler decision boundaries.

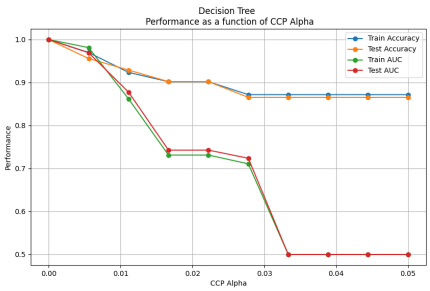

Figure 5: Auction Dataset: CCP Alpha Performance

In the dropout dataset, the "CCP_alpha" parameter was ad-

justed from 0.0 to 0.25 for the decision tree model. This time, we observed that as "CCP_alpha" increases, both the accuracies and the AUCs for the training and testing datasets follow the same trend and align closely. This indicates that the model is consistently balanced between being able to distinguish between classes (AUC) and making correct predictions (accuracy) across different levels of complexity. A likely explanation could be that the model, with increased generalization from a higher "CCP_alpha", maintains a stable performance, suggesting that the underlying data may have clear class separations or well-defined decision boundaries, making it easier for the model to predict accurately while also maintaining a good AUC.

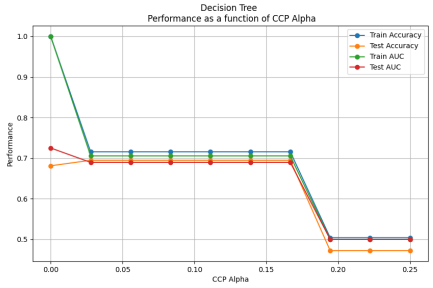

Figure 6: Dropout Dataset: CCP Alpha Performance

On the Auction dataset, we experimented by changing the maximum depth of the model from 1 to 20. Interestingly, throughout this range, the accuracy and AUC for both the training and test sets remained pretty similar. Moreover, as I increased the maximum depth, the performance values for both the training and test sets approached 1.0. This observation suggests that making the model fit the training set better by increasing the depth did not lead to overfitting on the test set.
It is unusual that increasing the max depth of the model would lead to an optimal fit of the both the training and the test set. It is important to narrow down why this might be. Since the dataset is taken from a trusted source, it is unlikely that this is a result of data leakage; moreover, since the default decision tree is not utilizing any form of regularization, it is unlikely that the model is simply regularizing well while maintaining outstanding performance. What is more likely is that the Auction dataset is sufficient to explain the phenomenon, and based on the initial exploratory data analysis, we can see that the dataset is discrete (non-continuous) in nature.

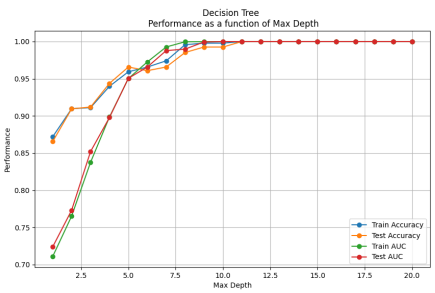

Figure 7: Auction Dataset: Max Depth Performance

For the Dropout dataset, it seems like when we make the decision tree more detailed by increasing the max depth, it gets better at recognizing patterns in the training set. However, when the depth goes beyond 5, the model starts to memorize the training data too much (overfitting), and its performance on new, unseen data (test set) starts to decrease in terms of both accuracy and AUC.

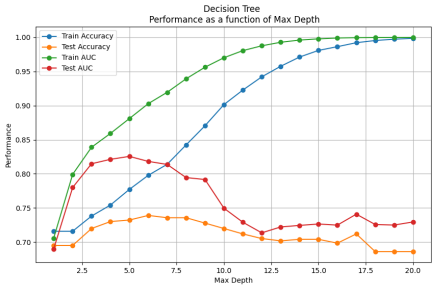

Figure 8: Dropout Dataset: Max Depth Performance

In the next experiment found in figures (9) and (10) for our decision tree classifier, we varied the value of the minimum impurity decrease hyperparameter from 0.0 to 0.005. The minimum impurity decrease is a measure calculated by selecting a split in a sample of points on a single dimension and evaluating then degree of disorder before and after doing so; if the disorder decreases substantially after a split, this split would have a high impurity decrease. The minimum impurity decrease can be thought of as a stopping condition for the decision tree branching strategy.
On the Auction dataset, increasing the minimum impurity decrease in turn decreases the variance of the model. By decrease the variance, our model underfits both the training and test set (since the dataset is sufficient to explain all phenomena).
For the Dropout dataset, the minimum impurity decrease of 0.001 achieve the optimal performance on the test set. After that, the test set performance stagnates as the training set performance decreases. This makes sense, as the higher impurity decrease prevents the training set from overfitting.

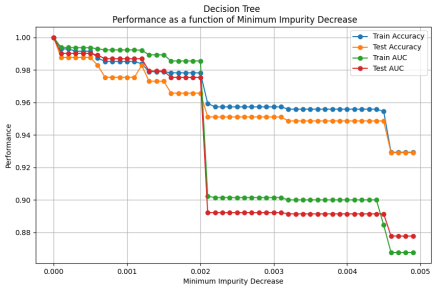

Figure 9: Auction Dataset: Minimum Impurity Decrease Performance

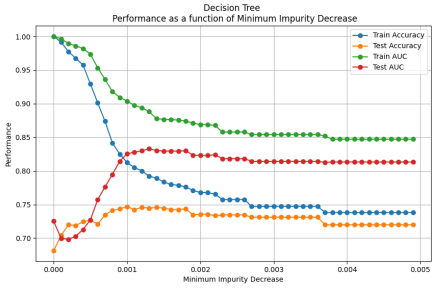

Figure 10: Dropout Dataset: Minimum Impurity Decrease Performance

We then performed experiments on minimum samples per leaf on the decision tree in figures (11) and (12). For the Auction dataset, As the minimum samples per leaf increases, the test and training accuracy and AUC drop linearly. Additionally, they drop at the same rate and also have nearly the same performance. This suggests that minimum samples per leaf provides generalization capabilities, allowing for the model to optimally fit the training and test set equally.
Moreover, the dropout dataset demonstrates a similar effect of bringing the training and test performances to nearly the same level at higher values of minmium samples per leaf.

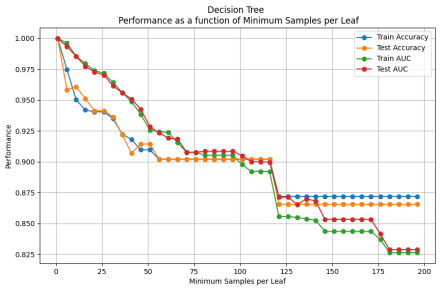

Figure 11: Auction Dataset: Minimum Samples per Leaf Performance

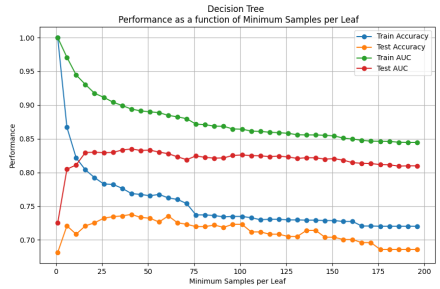

Figure 12: Dropout Dataset: Minimum Samples per Leaf Performance

From the decision boundaries seen in figure (13), it is evident that the decision tree model uses "splits" to separate the data. The decision tree uses simple splits across the dataset to arrive at a final model. This can be visualizes as the simple polygons and abrupt lines in the figure below. While the decision tree may generalize well and has an ease of explainability, it lacks variance and succumbs to a high degree of bias.

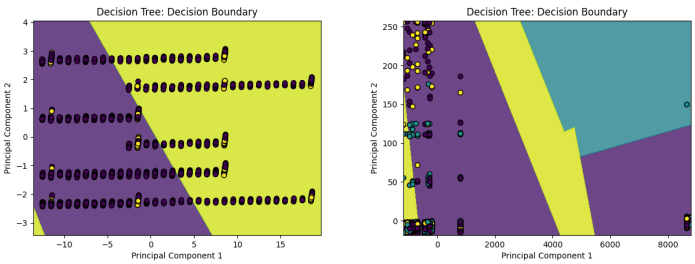

Figure 13: Auction Dataset (left) and Dropout Dataset (right): Decision Boundaries

3 Neural Network

We trained a neural network for 100 rounds (epochs) using different learning rates ranging from very small (1e-6) to very large (1e+2). It turns out that a learning rate of 1e-2 works best for both the training and the test sets in terms of accuracy and AUC, which are the same in this case. However, the training set always performs much better than the test set. Neural networks are particularly susceptible to overfitting on the training set. This neural network has a default dimension of (128, 64), and this neural network is shown to begin overfitting the Auction dataset on epoch 50 (Figure 19). We can see that the discrepancy between the test and training set can be attributed to this reason.

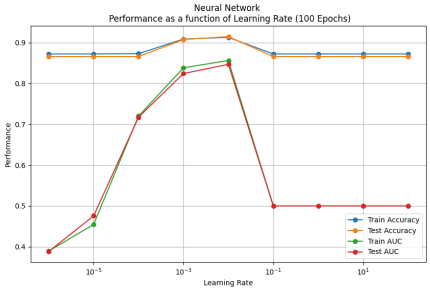

Figure 14: Auction Dataset: Learning Rate Performance

Additionally, we see a similar behaviour on the Dropout dataset, but the only difference is that the model performs best on learning rates two order of magnitude (1e-2) smaller. This suggests that the Dropout dataset has more complex relationships than the Auction dataset, requiring a lower learning rate. Moreover, the Dropout dataset has a larger scale on the input features (this dataset was not pre-processed), leading to a need for lower learning rates.

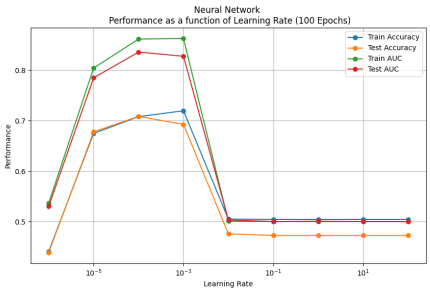

Figure 15: Dropout Dataset: Learning Rate Performance

We further explore hyperparameters of the neural network. In this experiment on the Auction dataset, we initialize the neural network to have a a learning rate of 1e-2, and then we very the hidden dimensions of the dataset. For the most part, the neural network performed well at all dimension combinations; there were a few dimension combinations, typically located at the graph extrema (triangle corners), which performed worse overall. This is likely due to the quadratic nature of multiplying the two dimensions, leading to a model that was slow to converge to a good solution. Generally, the best performing models has dimensional values in the middle ranges.

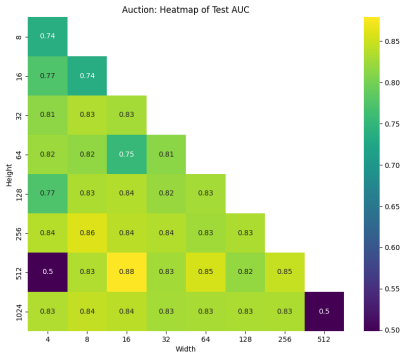

Figure 16: Auction Dataset: Dimension Performance Heatmap of Neural Network

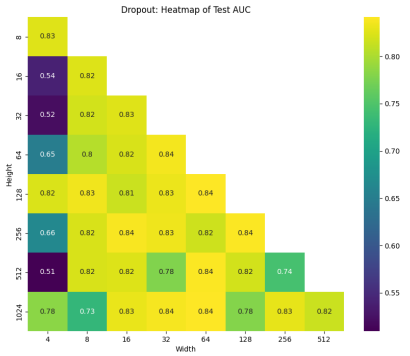

Figure 17: Dropout Dataset: Dimension Performance Heatmap of Neural Network

The below decision boundaries show the behavior of the neural network against the training set. We can not discern much from this figure except that the neural network shows smooth contour transitions from one classification region to then next; this smooth transition is likely due to the presence of non-linearities in the neural network such as the sigmoid as well as the sheer complexity of the model. Additionally, since the decision boundaries do not appear to map to the data, it can be said that the data lost in the principal component breakdown was necessary for explaining the model decisions.

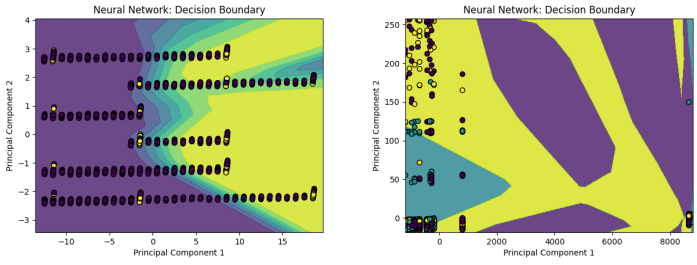

Figure 18: Auction Dataset (left) and Dropout Dataset (right): Decision Boundaries

Below we can see the performance curve of the neural network. This gives us an indication of at what epoch the model begins to overfit the training set, as this results in a marked decrease in performance on the validation set. In figure (19), we see that the neural network begins to overfit the training set of the Auction dataset at epoch 50, and in figure (20), we see that it begins to overfit the training set at epoch 75. In is notable to mention that in figure (19), despite the model overfitting the training set, it still continues to increase in performance on the validation set. It appears that despite the overfitting, the model retains generalizational capabilities, leading to an improvement on the validation set.

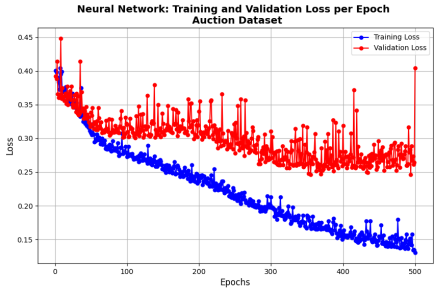

Figure 19: Auction Dataset: Training/Validation Curve per Epoch

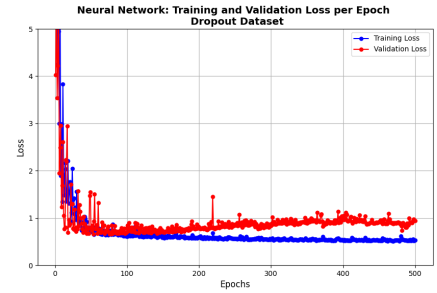

Figure 20: Dropout Dataset: Training/Validation Curve per Epoch

4 Boosting

We trained an XGBoost model using the default model parameters and varied multiple hyperparameters. For the first experiment seen in figure (21) and figure (22), the learning rate was varied from 0 to 1, and the XGBoost model was trained on both the Auction and the Dropout datasets. In the Auction dataset, a small learning rate led to a failed or slow convergence, resulting in poor performance across all benchmarks; but when the learning rate was of sufficient size, the model performed ideally. This behavior was different for the Dropout dataset, however. In the dropout dataset, the small learning rate had no adverse effect on the test set, but it does reduce performance on the training set. It appears that the XGBoost model performed about the same regardless of the learning rate on the test set metrics.
This behavior is comparable to the neural network where the Dropout dataset was optimal at lower learning rates; this behavior seems to hold slightly true for XGBoost as well.

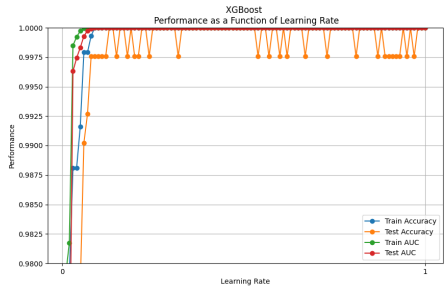

Figure 21: Auction Dataset: Learning Rate Performance of XGBoost

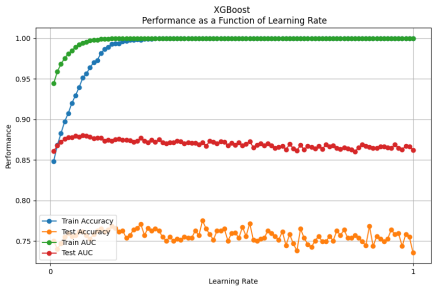

Figure 22: Dropout Dataset: Learning Rate Performance of XGBoost

In figure (22) and figure (23), we analyze the XGBoost model by varying the max depth hyperparameter from 1 to 10. This is akin to the same experiment where the max depth was varied for the decision tree. Just as before, it appears that the Auction dataset is sufficient to explain the phenomenon, so when optimally fitting the training set, the test set is fitted optimally as well. When the max depth of the model is at or below 5, the test metrics slightly underperforms the training metrics, giving credence to the idea that the Auction dataset is sufficient to explain itself.
Similar to the previous hyperparameter test, the max depth has no effect on the test set performance and only has an effect on the training set performance. Therefore, we should not consider these parameter when doing a grid hyperparameter search.

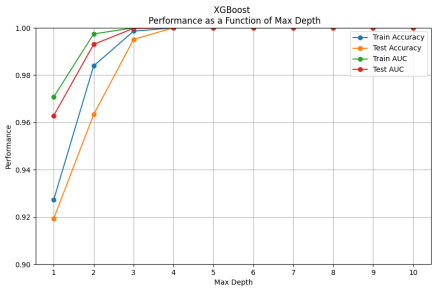

Figure 23: Auction Dataset: Max Depth Performance of XGBoost

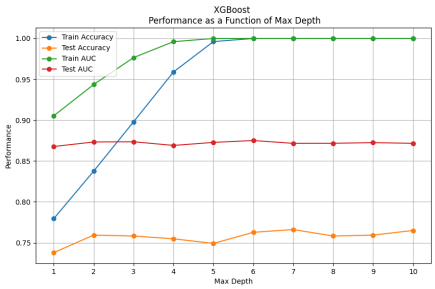

Figure 24: Dropout Dataset: Max Depth Performance of XGBoost

Yet again, when varying the N-estimators hyperparameter in the training of the XGBoost model in figure (25) and figure (26), we receive the same results as the previous two hyperparameters. The likely reason behind this is that increasing these hyperparamters straddles the bias/variance tradeoff of the model. Low values for each of these hyperparameters would represent a high degree of bias in the model, and high values would represent lower degrees of bias and more variance; this causes the model to overfit the training set. However, this does not explain why the test set performs well despite this increased variance; XGBoost has built-in regularization capabilities that allow the model to perform well on the test set. The XGBoost model incorporate both L1 (lasso) and L2 (ride) regularization into the model, increasing its effectiveness at high variance values.

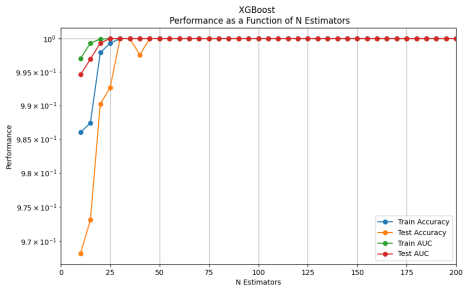

Figure 25: Auction Dataset: N-Estimators Performance of XGBoost

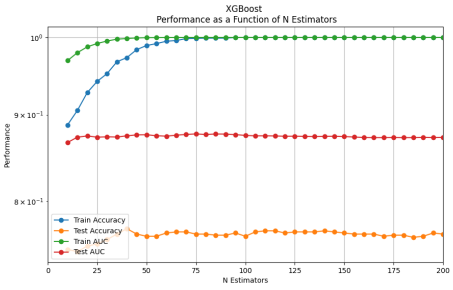

Figure 26: Dropout Dataset: N-Estimators Performance of XGBoost

Interestingly, the XGBoost model on the Auction dataset has decision boundaries when are heavily dependent upon the first principal component; it also has clear boundaries surrounding the majority of the yellow training set data, suggesting a more optimal fit. This is markedly different from the neural network decision boundary which has smooth transitions. This XGBoost model has more abrupt transitions in the decision bound-

aries, showing how the XGBoost model "splits" or "slices" the data. On the Dropout dataset, most of the decision boundary space is inconsequential to the true performance of the model; this is due to the outlier value in the bottom-right section of the graph. The decision boundary behavior appears comparable to the neural network behavior, but the boundaries are more jagged, beckoning back to the fact that the model "splits" the data.

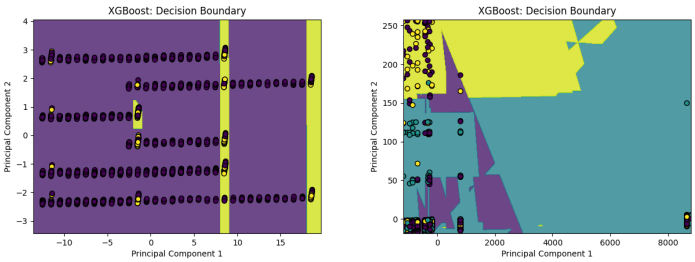

Figure 27: Auction Dataset (left) and Dropout Dataset (right): Decision Boundaries

5 Support Vector Machine

The ultimate goal of a support vector machine is to find the hyperplane which best separate the data. The support vector machine assumes that the data is linearly separable, which is not a greate assumption for these datasets. The support vectors refer to the data points that fall outside the decision boundary of the model. The kernel of the model performs the "kernel trick", which maps the high-dimensional input data to a linearly separable domain space.

In the below figures (28) and (29), we vary the values of the "Kernel" and the "C" parameter. The "C" parameter is a penalty term which determines the degree of penalty for misclassified results; in other words, it increases the variance of the model and decreases the bias.

Based on the results of figure (28) on the Auction dataset, the best Test AUC performance is achieved with an RBF kernel with a "C" parameter of 100; however, this is not the case for lower values of "C". This suggests that the RBF kernel fits the data well when the variance is high, simply being a characteristic of the dataset. Admittedly, this may be due to random chance (a happy coincidence of the exact test set chosen). Moreover, the polynomial kernel consistently performs well on the test AUC performance, suggesting that the polynomial kernel may have the most reproducible results. Straggling behind slightly, the linear kernel also seems to fit the data relatively well, likely due to its simplicity of assumptions (linear bias).

In figure (29), the Dropout dataset performs better with a linear kernel, and the "C" parameter of 0.1 maximizes the test AUC. This is a very interesting result, suggesting that the most biases of the kernels (linear and 0.1 "C" parameter) is the most performant. This could mean that the kernels and "C" parameters promoting high variance are quickly overfitting the training set, resulting in poor performance on the test set; therefore, the most generalizable models are the ones with a higher degree of bias.

Kernel	C Parameter	Train Accuracy	Test Accuracy	Train AUC	Test AUC
linear	0.1	0.871938	0.865526	0.578082	0.605701
linear	1	0.871938	0.865526	0.753984	0.762866
linear	10	0.871938	0.865526	0.72271	0.730611
linear	100	0.871938	0.865526	0.743288	0.756549
poly	0.1	0.871938	0.865526	0.71247	0.721315
poly	1	0.871938	0.865526	0.717426	0.731279
poly	10	0.871938	0.865526	0.807699	0.797432
poly	100	0.871938	0.865526	0.810673	0.789111
rbf	0.1	0.871938	0.865526	0.596874	0.660606
rbf	1	0.871938	0.865526	0.5313	0.450539
rbf	10	0.871938	0.865526	0.611781	0.608937
rbf	100	0.871938	0.865526	0.837162	0.812583
sigmoid	0.1	0.871938	0.865526	0.675618	0.688238
sigmoid	1	0.876137	0.860636	0.675618	0.688238
sigmoid	10	0.825052	0.809291	0.668636	0.681921
sigmoid	100	0.815255	0.801956	0.66818	0.681715

Figure 28: Auction Dataset: Kernel Performance of SVM

Kernel	C Parameter	Train Accuracy	Test Accuracy	Train AUC	Test AUC
linear	0.1	0.775194	0.742373	0.84835	0.811198
linear	1	0.776163	0.754802	0.837259	0.806265
linear	10	0.776486	0.742373	0.843446	0.805927
linear	100	0.772287	0.746893	0.834045	0.803632
poly	0.1	0.504199	0.472316	0.730617	0.728968
poly	1	0.504199	0.472316	0.767605	0.765184
poly	10	0.504199	0.472316	0.772623	0.770234
poly	100	0.585271	0.554802	0.758341	0.763789
rbf	0.1	0.504199	0.472316	0.729491	0.694282
rbf	1	0.504199	0.472316	0.782106	0.751495
rbf	10	0.504199	0.472316	0.786327	0.770245
rbf	100	0.504845	0.472316	0.786634	0.765886
sigmoid	0.1	0.504199	0.472316	0.508672	0.462256
sigmoid	1	0.491279	0.465537	0.526527	0.482229
sigmoid	10	0.405039	0.439548	0.574342	0.555079
sigmoid	100	0.401486	0.430508	0.576819	0.559985

Figure 29: Dropout Dataset: Kernel Performance of SVM

This analysis of the "C" parameter also holds true on figures (30) and (31). These figures further support the analysis that on the Auction dataset, larger "C" parameters increase performance on the test set. Additionally, on the Dropout dataset, the model tends to reverse this trend and perform better at lower "C" values. It is important to note that the support vector machine is among the worst performing models analyzed in these experiments. Futhermore, on figure (31), the high "C" paramter completely removes the models capability of generalizing on the test set, and there is a slight performance decrease on then training set; this may be a result of the decision boundary being overly sensitive to particular data samples in the training set, fitting those more heavily and underfitting another subset of the training set; this also has an adverse effect on the test set performance.

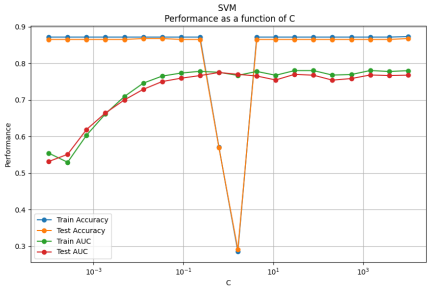

Figure 30: Auction Dataset: Performance as a Function of C

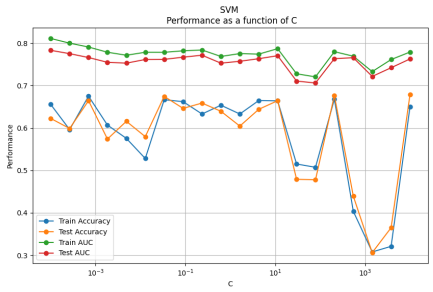

Figure 31: Dropout Dataset: Performance as a Function of C

The below decision boundaries accentuate the "single boundary" nature of a support vector machine. On the left graph, there is a single boundary visible, and this boundary implies that the support vector machine has a bias towards a single class. This could explain the poor AUC performance on the test set while still retaining an okay accuracy.
The right graph - the Dropout dataset - shows three decision boundaries. This is because a single support vector machine does not have multi-class capabilities; instead, it must resort to the one-vs-rest (ovr) multi-class strategy. The "ovr" multi-class strategy was shown to outperform the "crammer singer" multi-class strategy in randomized grid search.

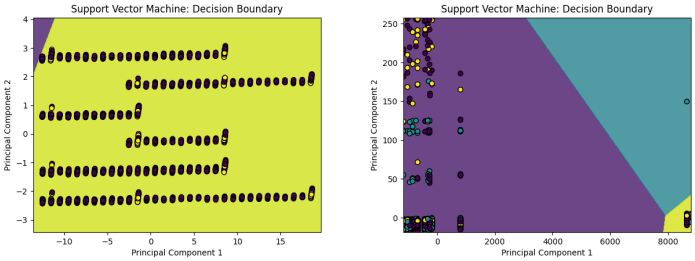

Figure 32: Auction Dataset (left) and Dropout Dataset (right): Decision Boundaries

6 K-Nearest Neighbors

The k-Nearest Neighbors (kNN) model is perhaps the most heavy-weight model of all analyzed in this project. This is because of the fact that the kNN model is an instance-based or memory-based model. These instance-based models often have the largest memory requirements.
We varied the values of k in both the Auction and Dropouts datasets in figures (33) and (34). Unsurprisingly, the Auction dataset showed a mirrored decrease in performance on both the test and training set as a function of "k". Yet again, this shows the sufficiency of the dataset to explain its own internal phenomena.
In figure (34), the kNN performs poorly overall, with a maximum test AUC of approximately 0.7. The reason this may be occurring is because of the lack of pre-processing of the Dropout dataset. The Dropout dataset has un-scaled features which are dominating the model outcomes. In short, the model's reliance on euclidean distance metrics prevents it from performing well without a scaling pre-processing step such as StandardScaler.

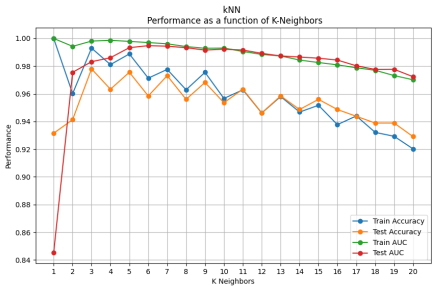

Figure 33: Auction Dataset: Performance as a Function of K-Neighbors

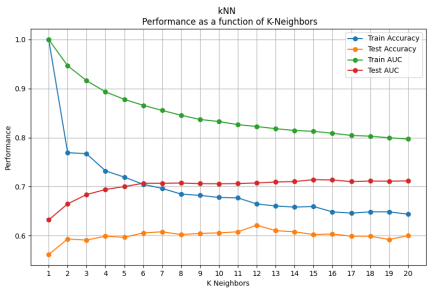

Figure 34: Dropout Dataset: Performance as a Function of K-Neighbors

This is easily apparent in the Dropout SHAP output for the kNN model found in figure (35). The "Course" feature dominates the kNN result. However, the "Course" feature appears to be a large float value, but it is actually a float categorical value representing a class. This results in the kNN overfitting the "Course" feature greatly, and failing to generalize. In future work, this "Course" feature must be pre-processes by being categorically encoded.

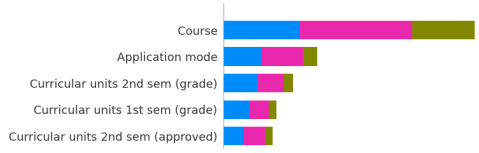

Figure 35: Dropout Dataset: kNN SHAP Output of Top 5 Features

After pre-processing the Dropout dataset by one-hot encoding the "Course" feature and then using StandardScaler fitted on the training set, the kNN was made capable of outperforming the even the XGBoost model. In figure (36), we see that the kNN model acheived a peak test AUC of 0.9895 at a k-neighbors parameter of 2. This nearly perfect performance exceeds that of the XGBoost model seen in figure (39). This shows the effect of scaling the Dropout dataset. Not only did every metric perform better, but we also see an interesting downtrend in performance as the k value was increased. This suggests that the additional model bias that comes from using more than 4 neighbors only serves to reduce training set performance, meaning that higher variance in this instance is better than higher bias. This also means that the Dropout dataset is sufficient to explain itself - same as the Auction dataset.

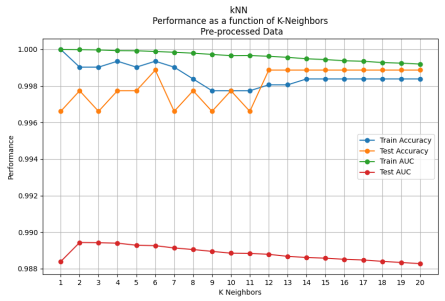

Figure 36: Pre-Processed Dropout Dataset: Performance as a Function of K-Neighbors

After pre-processing the Dropout dataset before using it to train the kNN model with k-fold cross validation and randomized grid search, the model outperformed even XGBoost (as seen in figure (37)). Interestingly, while the test accuracy is lower than the training accuracy, the test AUC exceeds the training AUC. This suggests that the training set has slightly more imbalanced performance than the test set.

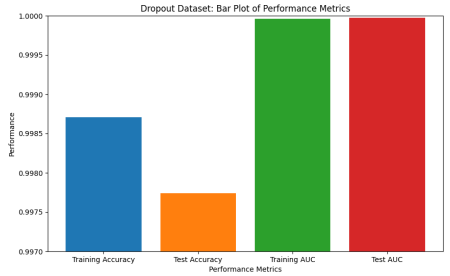

Figure 37: Pre-Processed Dropout Dataset: Overall Performance of Model Trained with k-fold Cross Validation and Randomized Search

The Euclidean nature of the kNN model is on full display in figure (38). The decision boundaries of the model look almost like torn paper or a stain. The decision boundary of the kNN model is as detailed as the dataset itself, so for every variation in the dataset, there is an equal level of variation in the kNN decision boundary.

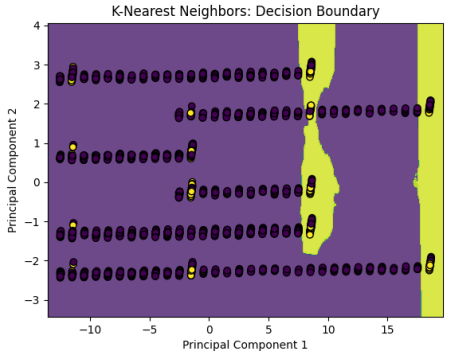

Figure 38: Auction Dataset: KNN Decision Boundary

7 Feature Importance

Below in figures (39) and (40), we see the the feature importance SHAP values for XGBoost. These values were used to generate the exploratory data analysis graphs earlier in the paper within the first section. The results of these graphs were used iteratively to circle back to the datasets and examine their properties which led to their significance here.

Figure 39: Auction Dataset: SHAP Feature Importance of XGBoost

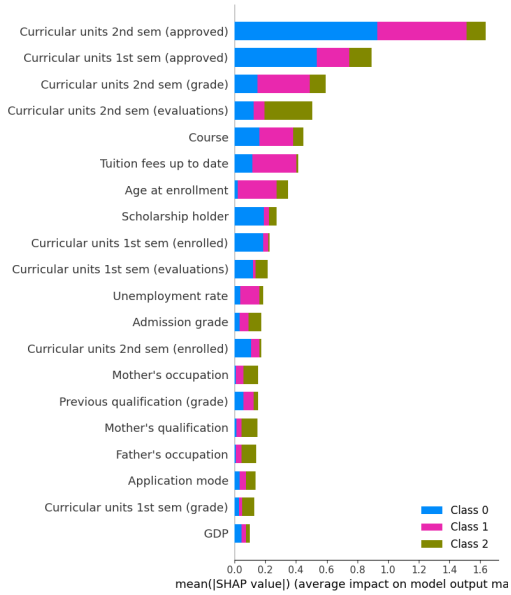

Figure 40: Dropout Dataset: SHAP Feature Importance of XGBoost

8 Performance

The below performance tables of figures (41) and (42) are the final performance values of the Auction dataset and Dropout dataset, respectively. It should be noted that the pre-processed kNN performance is not represented in this table; for this information, reference figure (37) found earlier.

These tables were created using randomized grid search with 5-fold cross validation. The number of iterations used was model-specific, as some models were time-prohibitive.

The XGBoost model was always among the best performing models on both dataset, with it being beaten out by kNN on the dropout dataset.

Among all of the models, the support vector machine had the slowest time-to-train per iteration. This makes sense, as the kernel trick is memory intensive. Additionally, the most complex model - XGBoost - had the next highest time-to-train per iteration. It was difficult to compare this to the neural network due to the neural network being trained with static hyperparameters on 500 epochs. For the purpose of the graph, an epoch was considered an iteration.

Model	Test Accuracy	Test AUC	Train Accuracy	Train AUC	Time to Train (TTT) [seconds / iteration]
Decision Tree	0.804401	0.811145	0.789363	0.792981	0.0327955
K-Nearest Neighbors	0.982885	0.998254	1	1	0.0272502
Neural Network	0.909535	0.929327	0.942617	0.964569	0.218392
Support Vector Machine	0.865526	0.769902	0.871938	0.781421	12.7149
XGBoost	0.99022	0.998305	0.9993	0.999754	0.449226

Figure 41: Auction Dataset: K-Fold Cross Validation Performance with Randomized Grid Search Performance Comparison

Model	Test Accuracy	Test AUC	Train Accuracy	Train AUC	Time to Train (TTT) [seconds / iteration]
Decision Tree	0.718644	0.816825	0.729651	0.840681	0.0124001
K-Nearest Neighbors	0.641808	0.772328	1	1	0.0499408
Neural Network	0.728814	0.835307	0.784238	0.899247	0.193511
Support Vector Machine	0.753672	0.853605	0.774548	0.886188	37.9302
XGBoost	0.754802	0.878453	0.84916	0.953311	3.83413

Figure 42: Dropout Dataset: K-Fold Cross Validation Performance with Randomized Grid Search Performance Comparison; NOTE: This figure was made before performing the StandardScaler and categorical encoding pre-processing adjustment going into the kNN

In figures (43) to (47), we can observe the distinct performance curves associated with each model. Essentially, a performance curve provides a visual representation of how well a model is learning as it is exposed to more of the training set, measured by the percentage of the training set that the learner has seen. To elaborate, for this specific experiment, we conducted a total of 10 trials. In each of these trials, we incrementally added 10% more data to the training set. Following the addition of data in each trial, we evaluated the performance of every learner on the holdout/test set.

One of the key observations was made in figure (46). Here, we noticed that the performance of the Support Vector Machine (SVM) on the Auction dataset remained relatively unchanged after the model had seen 30% of the training set. This stationary behavior of the SVM implies that it might not be generalizing as effectively to the test data as one would hope. Contrarily, both kNN (k-Nearest Neighbors) and XGBoost displayed a linear uptrend in performance in response to the same increase in exposure to the training set. It's noteworthy that this generalization issue wasn't exclusive to the SVM; both the Neural Network and Decision Tree models exhibited similar behavior.

Interestingly enough, a parallel can be drawn between the performance curves of the XGBoost model and kNN. Their curves mirrored each other as the percentage of the training set seen increased. This resemblance in performance progression could be attributed to the complexity of the XGBoost model, which is comparable to that of the kNN model. The inherent complexity and the high variance of both models might be the reason they effectively integrate new training set data, consequently enhancing their performance on the test set.

This alignment in performance curves suggests that both the XGBoost and kNN models have a similar capacity for learning and adapting. The learning dynamics shown by these two models indicates that they can efficiently assimilate additional information from the training set, translating it into improved performance when faced with unseen data.

In conclusion, the observations derived from Figures (43) through (47) offer valuable insights into the learning capabilities of different models. The stagnation in the performance of the SVM, Neural Network, and Decision Tree post 30% exposure to the training set raises questions about their ability to generalize effectively. In stark contrast, the aligned and steadily improving performance of the kNN and XGBoost models underscores their adaptability and potential for effectively handling unseen data.

Figure 43: Auction Dataset (left) and Dropout Dataset (right); Performance as a Percentage of the Seen Training Set

Figure 44: Auction Dataset (left) and Dropout Dataset (right); Performance as a Percentage of the Seen Training Set

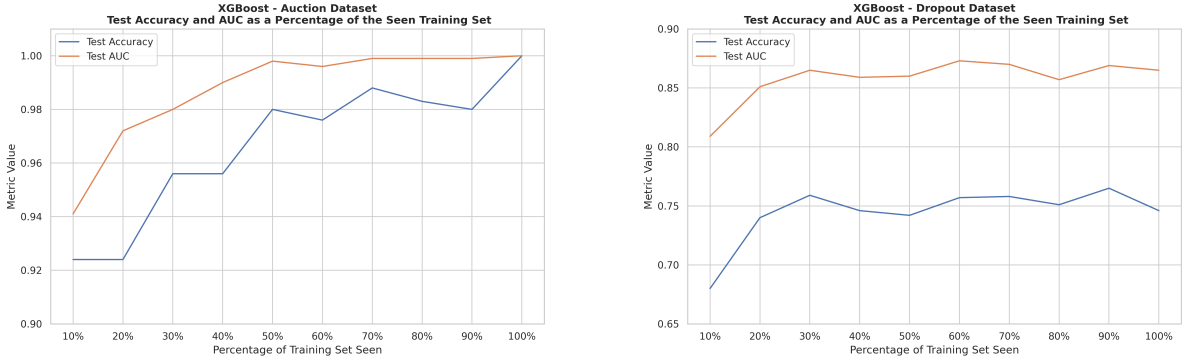

Figure 45: Auction Dataset (left) and Dropout Dataset (right); Performance as a Percentage of the Seen Training Set

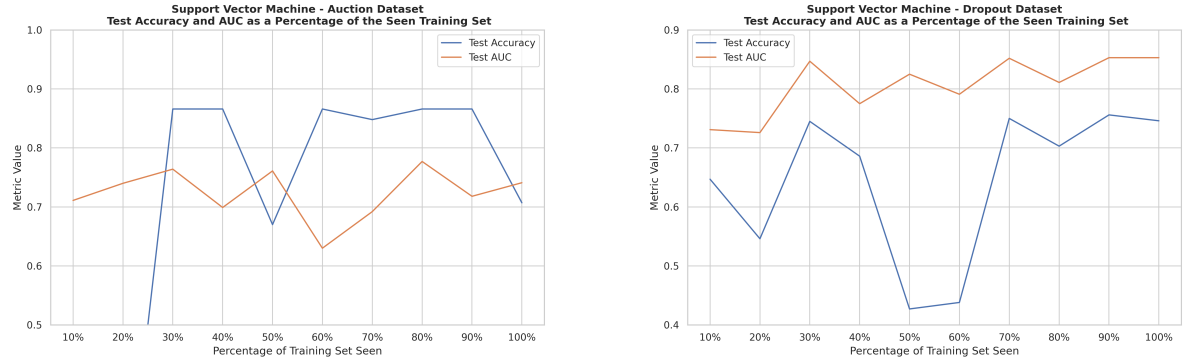

Figure 46: Auction Dataset (left) and Dropout Dataset (right); Performance as a Percentage of the Seen Training Set

Figure 47: Auction Dataset (left) and Dropout Dataset (right); Performance as a Percentage of the Seen Training Set

9 Conclusion

Auction verification and detecting undesirable auction outcomes are pivotal in addressing revenue loss, with a specialized dataset sourced from the UC Irvine Machine Learning Repository employed to analyze bidder behaviors and auction outcomes. This dataset, consisting of 2043 rows after duplicate removal, was approached with a classification target in mind, revealing non-linear correlations between bidder capacities and auction specifics. Additionally, a student dropout dataset containing various student-related features such as academic performance and demographics was utilized to predict student outcomes through a classification model, with the second semester features being particularly indicative of academic success. The decision tree models, applied to both datasets, were fine-tuned using various hyperparameters like "CCP_alpha", maximum depth, and minimum impurity decrease. The experiments demonstrated that increased model complexity and depth did not necessarily lead to overfitting, with some instances revealing optimal fit for both training and test sets. The models' adaptability and performance across datasets suggest a balance in distinguishing between classes and maintaining accurate predictions, highlighting the discrete nature of the auction dataset and the nuanced decision boundaries within the dropout dataset.

In this paper, various models were applied to the Auction and Dropout datasets, focusing on the effects of modifying hyperparameters. For decision trees, adjusting the minimum impurity decrease and minimum samples per leaf showcased their influence on model variance and generalization. Neural networks, with different learning rates and dimensions, revealed tendencies to overfit, especially evident around epoch 50 for the Auction dataset. A learning rate of 1e-2 was found optimal, although Dropout dataset demanded a smaller learning rate, hinting at its complexity. XGBoost models displayed distinctive responses to learning rate variations, max depth, and N-estimators on the two datasets, highlighting the robustness of XGBoost due to built-in regularization. The comparison of decision boundaries between neural networks and XGBoost models revealed smooth transitions for the former and abrupt, split-like transitions for the latter, indicating different fitting and generalization characteristics. The findings underline the significance of hyperparameter tuning and the inherent traits of different models in handling datasets with varied complexities.

In experiments, different models and hyperparameters were evaluated on Auction and Dropout datasets. The decision tree model showed underfitting with increased minimum impurity decrease, while various minimum samples per leaf provided generalization capabilities. The neural network, with optimal learning rate at 1e-2, exhibited overfitting from epoch 50, and varied performance with hidden dimensions, emphasizing smooth decision boundaries. XGBoost models were trained with varied learning rates, max depths, and N-estimators, revealing different behaviors and decision boundaries for the two datasets and highlighting the model's regularization capabilities. For the Support Vector Machine, the "C" parameter and kernel type significantly influenced performance, with higher "C" values benefiting the Auction dataset and lower values favoring the Dropout dataset. The SVM depicted single and multiple decision boundaries, reflecting its bias towards specific classes. The k-Nearest Neighbors (kNN) model, highly memory-intensive, revealed a mirrored decrease in performance with increasing "k" and significant reliance on feature scaling, which, when properly executed, led to superior performance, surpassing even XGBoost. The results underscore the need for appropriate preprocessing and the datasets' ability to explain themselves.

Through k-fold cross-validation and randomized grid search, the pre-processed kNN model displayed superior performance over other models, including XGBoost, on the Dropout dataset, despite exhibiting more nuanced decision boundaries. Performance tables highlighted the consistent efficacy of XGBoost and the slower training times of support vector machines and XGBoost. Notably, performance curves revealed that while SVM, Neural Network, and Decision Tree models exhibited stagnant performance after viewing 30% of the training data, XGBoost and kNN showcased aligned, steadily improving performance with increased exposure to the training set, indicating their adaptability and efficient learning capacity. These insights raise questions about the generalization capabilities of some models while emphasizing the promising adaptability of kNN and XGBoost in assimilating new information for effective performance on unseen data.