ASSIGNMENT 1: SUPERVISED LEARNING

Casey Hirschmann (chirschmann3@gatech.edu)

1 Intro

The selected datasets are 'Adult' and 'Mushrooms', referred to as 'Salary' and 'Shrooms' moving forward (UCI ML Repository). Data information is summarized in Table 1. Missing instances were removed in 'Salary' as there was enough data to train the models (seen in later learning curves) to avoid imputation. One feature was removed from 'Shrooms' due to the amount of missing data – satisfactory results were achievable without this feature. All categorical data was one-hot encoded before running algorithms.

Table 1: Data set summary

Dataset	# Instances	# Features	Encoding
Salary	32,561 (30,162 post-cleaning)	14	0: <=$50k 1: >$50k
Shrooms	8124	22 (21 post-cleaning)	0: Poisonous 1: Edible

1.1 *Salary data*

'Salary' is an interesting dataset due to three components: the data is heavily imbalanced (75% <=$50k), many features are not normally distributed, and ethical issues arise due to the use of certain protected class variables (sex, race, etc.).

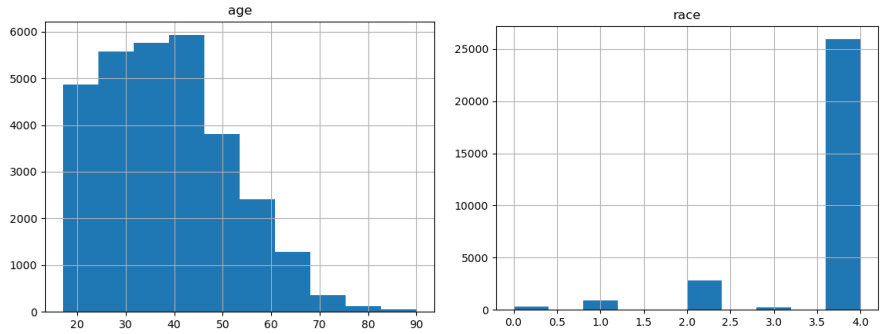

Figure 1: Histograms of 'Salary' data. (1a) non-normally distributed age. (1b) Sparse instances of certain races.

Heavily imbalanced data leads to a misleadingly high accuracy score with the algorithms since, off the bat, it could get 75% correct simply by guessing the more heavily present class. F1 scores revealed much lower performance. SMOTE oversampling was used to deal with this. Oversampling was chosen to avoid induced bias via undersampling. Figure 1b shows there are many cases of sparse data that could randomly be removed via undersampling. It is recommended to use under *and* over sampling, however I am only human. Due to this laziness, some models are more likely to overfit due to introduced repeated data in the sample (2U, inc., 2022). SMOTE was chosen because it was the "most basic" method. This choice however disadvantages "edge case" minority examples where other methods (ADASYN, Borderline-SMOTE SVM, etc.) focus on "low density" areas of data (Kavish, 2022). Employing one of these methods + undersampling would further decrease error on many of the algorithms. Note the accuracy metric chosen moving forward is F1 to account for any issues with the unbalanced data although accuracy and F1 mostly converged post-oversampling.

Figure 1a shows one of the features that isn't normally distributed. This causes issues when doing a train/test split as the sample is less likely to represent the population and lead to higher variance between training and testing data.

Combining the sparcity of certain data (see lack of race variety in Figure 1b) and the imbalance of data, all algorithms will be ripe with bias towards particular races and genders (i.e. likely to have high error for the non-dominant class). Usage of any of these algorithms for making life-impacting decisions (loan approval, criminality, etc.) would be highly unethical as certain protected classes would likely be unfairly impacted.

1.2 *Shrooms data*

'Shrooms' is inherently boring at first glance: perfect performance in most algorithms, balanced data, etc. However, it shan't be written off so easily. 'Shrooms' provides us insight into handling of edge cases with sparse data, the impact of tuning hyperparameters, and a glimpse into the 'why' behind this data set's superior performance and 'Salary's' lesser performance in most algorithms.

'Shrooms' data is what can only be describe as "whack". 'Salary' has some edge cases, but 'Shrooms' takes this to another level, as seen in Figure 2. There is a profusion of very sparse data. However, it still performs MUCH better than 'Salary'. As we will see, much

of this lies in the difinitiveness of certain features where 'Salary' has less obvious separability in the data. However, these 'Shroom' sporadic warriors still creep in and cause some error as we will see.

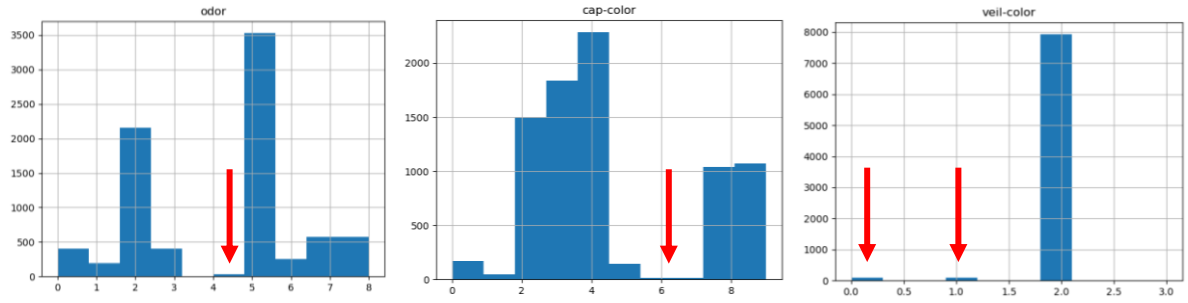

Figure 2: Histograms of some of the (many) Shrooms features showing sparse data.

2.3 *Splitting Data*

Data was split into an 80/20 split between train and test. This was further split into an 80/20 split for a training/validation set to use for the SVM confusion matrices and the cross-validation in the ANNs. All other algorithms had cross-validation built in for creating validation and loss curves. Test data wasn't used until the final learning curves build and accuracy comparison. 80/20 was arbitrarily picked to ensure there was plenty of data for training. This lead to some "chaotic" results in the ANN neural networks since the testing data was slim.

2 Overall Results

2.1 *F1 and Accuracy*

Table 2: Algorithm results comparison and optimal hyperparameters.

	Decision Trees		**XGBoost**		**KNN**		**SVM**				**ANN**	
	Salary	Shroom	Salary	Shrooms	Salary	Shroom	Salary		Shrooms		Salary	Shroom
Accrcy	0.872	0.999	0.906	1.0	0.876	1.0	0.899	0.896	1.0	1.0	0.892	1.0
F1	0.869	0.999	0.905	1.0	0.877	1.0	0.897	0.895	1.0	1.0	-	-
Best Param	**Max Dep**: 14 **Min Samps Leaf**: 12	**Max Dep**: 6 **Min Samps Leaf**: 4	**Col Samp by Tree**: 0.8 **Gamma**: 1.5 **Max Dep**: 6 **# Estimtrs**: 60	**Col Samp by Tree**: 0.6 **Gamma**: 0.5 **Max Dep**: 6 **# Estimtrs**: 15	**k**: 15 **weight**: uniform	**k**: 5 **weight**: uniform	**Kern**: RBF **C**: 1	**Kern**: Poly **C**: 1	**Kern**: RBF **C**: 1	**Kern**: Poly **C**: 1	**# Layers**: 1 **Learn Rate**: 0.01	**# Layers**: 1 **Learn Rate**: 0.01

The similarity of the F1 and accuracy scores shows that the data oversampling did its job. Because of that, ANN only tracked accuracy for time constraint issues (and by time constraint I mean I didn't have time to figure out how to produce F1 in Keras…).

The "best" performer from a pure accuracy perspective is XGBoost for Salary and really anything for Shrooms. It makes sense that Boost performs slightly better than DT since it is places emphasis on mistakes (i.e. some of these strange outliers from the non-normally distributed data). This also explains why ANN performs relatively well since it can capture some of the complexities of the data. SVM can also capture this, especially with RBF which is acting in an infinite dimension. Inversely, KNN is more likely to be skewed by some of these outliers, and thus performs slightly worse.

2.2 *Time*

Table 3: Algorithm time comparison.

	Decision Trees		**XGBoost**		**KNN**		**SVM - RBF**		**SVM - Poly**		**ANN**	
Time (secs)	Build	Query	Build	Query	Build	Query	Build	Query	Build	Query	Build	Query
Salary	0.26	0.01	1.77	0.02	0.02	1.09	49.81	10.84	52.05	4.44	137.53	1.62
Shrooms	0.02	0.001	0.11	0.01	0.01	0.15	0.43	0.16	0.33	0.04	26.31	0.57

From a pure timing perspective, DTs are the obvious winner here. There is a bit of a trade off between build and query time when compared to the runner up, KNN, but it is still overall faster. KNN will obviously be the fastest build as it has to learn nothing and only save the data which also explains the lengthier query process. DTs, along with XGBoost, have fast query, because it is easy to follow the branches of the tree. XGBoost is slightly longer to build than DTs since many small trees are required to train.

SVMS take a large time step up from the prior three due to the "curse of dimensionality" with the number of features impacting the multi-dimensional space (Bengio et al., 2005, 1). This also explains why Salary takes longer than Shrooms since there seem to be many more irrelevant features in Salary that complicate the space and the determination of the margins.

Finally, ANN is DRASTICALLY worse than any of the previous due to the feedback loop required to train through multiple epochs. Cutting down epochs would improve this, but the data risks underfitting.

2.3 *Best Overall*

Best overall algorithm is a balancing of time, accuracy, and ability to update the model. The learning curves (below sections) show that there is plenty of data to train both algorithms, so the ability to update as new data comes in isn't as much of a concern, thus, query time is more important than build time. XGBoost is likely the best algorithm for Shrooms and Salary (best accuracy and query time combo). Salary's close second in accuracy – RBF Kernel SVM – takes far too long to build and query to be a contender for the minimal accuracy improvement over KNN or DT.

3 Decision Trees

3.1 *Bias vs Variance*

Decision trees typically have low-bias and high variance (Brownlee, 2016). However, the final 'Salary' tree (Figure 3a) shows lower variance and higher bias. This is due to pruning. Figure 4 shows the validation curves for the two chosen hyperparameters (max depth and min samples per leaf). The best parameters determined through the grid search (max depth=14, min samples=12) align with the area where each hyper parameter begins to overfit (moving right in max depth and left in min samples). Note these validation curves were created by utilizing auto values for Scikit's DecisionTreeClassifier outside of the hyperparameter of interest.

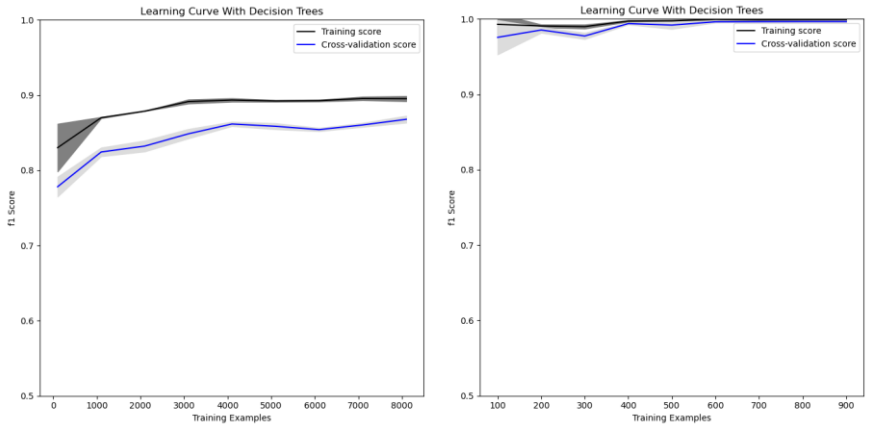

Figure 3: Final learning curves using F1 score for (a - left) Salary data and (b - right) Shrooms data.

If the tree were grown to these higher depths or lower samples per leaf, the variance in the algorithm would greatly increase and the training F1 in the final model would approach 1.0.

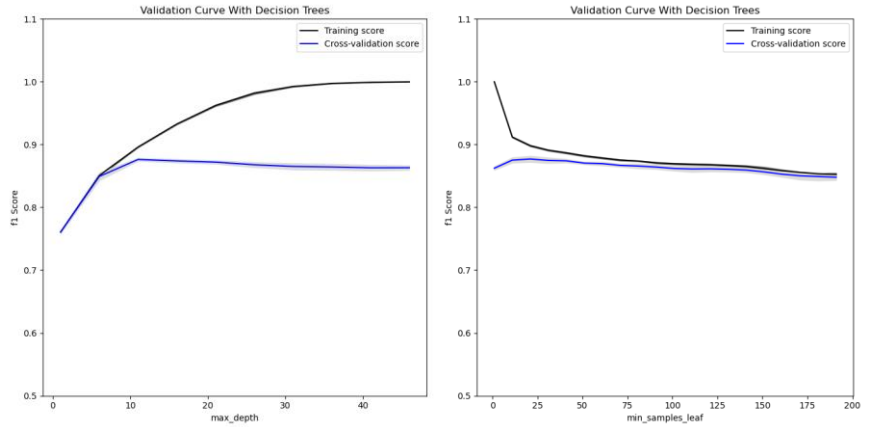

Figure 4: Validation curves for Salary data investigating (left) max tree depth and (right) minimum samples per leaf.

It is worth noting that the learning curves reveal that more data won't improve the F1 score for either algorithm as they both plateau. It is also interesting to note that the training model (not just CV) gets worse with less data (vs other algorithms where training score decreases with increased data like Boosting). This is also due to the pruning – forcing a halt at 14 samples per leaf when one has so few data points will yield higher error since you are likely to have less "similar" data points and are thus forcing higher Gini index/entropy at the terminal leaf.

3.2 *Gini vs entropy*

There is minimal difference in these splitting parameters outside of the time to perform the training (Aznar, 2020). This matters for these two algorithms as 'Salary' is a *much* larger tree than 'Shrooms', so Gini is used for 'Salary' and held constant for 'Shrooms'. Note this saved 'Salary' 0.01 secs, so not significant, but over larger data sets could matter.

3.3 *Salary vs Shrooms and Improving Performance*

Now back to the promise of convincing ourselves 'Shrooms' is interesting.

For point 1, we look at the final decision trees. Figure 5 shows us a view of the trees compared – do not try to read what they say, this is not the point. Shrooms has many more features than Salary, yet its tree is much smaller, and many terminal nodes are able to achieve 0.0 entropy long before the min samples per leaf is met (trust me on this since you can't read it, but one leaf has 2619 samples). This shows that there are a few features that are strong indicators of the class for Shrooms versus the Salary data that shows classification is a bit more complicated. This could indicate that eliminating some of the unnecessary features in Salary may help to improve results. Moral of the story, fewer features don't mean higher F1 scores – the importance of your feature is what matters.

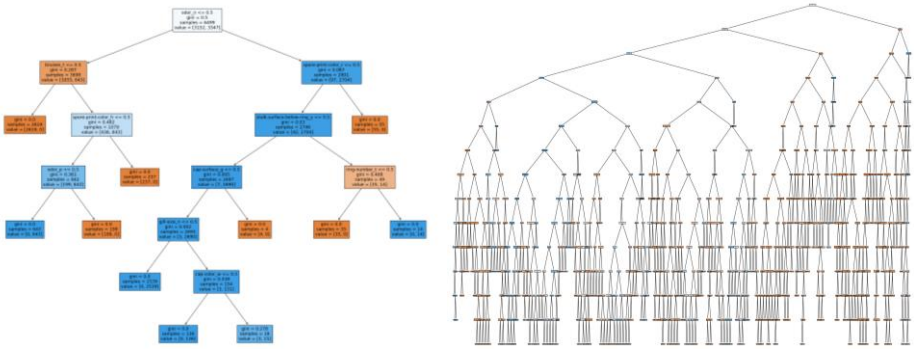

Figure 5: Final decision trees for (left) Shrooms and (right) Salary.

For point 2, DT is the only algorithm that Shrooms can't get to 100% accuracy. This is because the only leaf without Gini=0.0 has a 3, 15 split of data. The data is an edge case where it would need multiple more layers to split this data which reeks of overfitting. Although the LC shows it can reach 100% F1 with enough data, the topic of this data isn't one for overfitting (you don't want to eat a poisonous mushroom because someone wanted to force 100% accuracy which overfits the model).

3.4 *Feature Importance*

To further reiterate how few important features and the severity of importance of Shroom's features, see Figure 6 (again, reading labels isn't the point here so please don't). You can see mushroom odor is aggressively important with few other features trailing that. Salary has a closer competition between its features with many more contenders in the race. This trend will continue through all algorithms. Note the reason there are a billion features is due to the one-hot encoding performed.

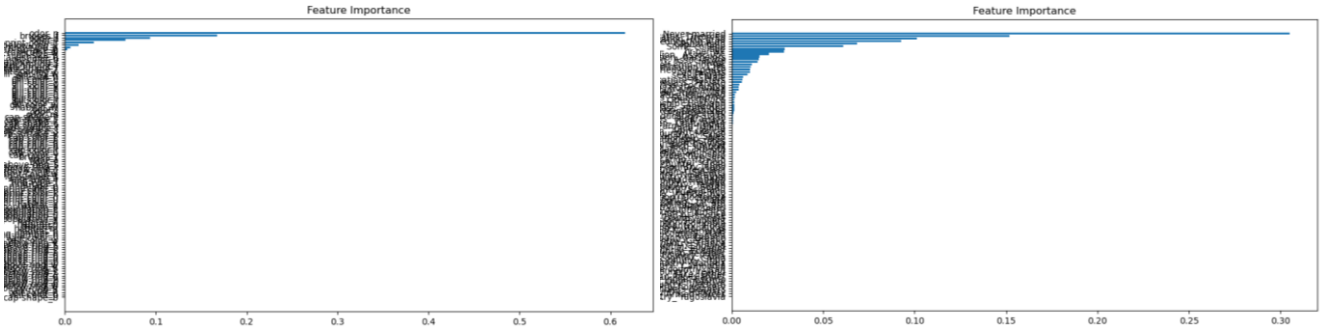

Figure 6: Feature importance for (left) Shrooms and (right) Salary data with the final DT.

4 XGBoost

4.1 *Why XGBoost*

Adaboost is inferior to XGBoost when dealing with insignificant features (Nikulski, 2020). Since the DT for 'Salary' implied we have some irrelevant features, XGBoost became the superior choice. XGBoost is also crafted for speed which is important since we are already pretty accurate with DTs (too much added time may not be worth the tradeoff).

4.2 *Salary's Boost*

The final learning curve (LC) (Figure 7a) looks like the DT LC with slightly less bias since this algorithm is essentially a combination of multiple "weak" DTs. The interesting aspect here is comparing the final LC to the intermediate LC (Figure 7b). The intermediate LC is fit using auto values except for max depth and number of estimators which were determined using validation curves. The grid search had other hyperparameters fed in to optimize which led to a different final LC. This shows that overfitting mitigation with XGBoost requires the tuning of multiple hyperparameters.

This is further explained with the validation curves (Figure 8) for the two hyperparameters. The 'depth of tree' hyperparameter shows a higher propensity for overfit than 'number of estimators'. 'Number of estimators' will begin yielding diminishing returns after a point since it can no longer correct the errors it is trying to improve; thus, overfitting isn't the concern here. However, more depth ⇒ larger trees ⇒ capturing more detail ⇒ overfitting potential (Brownlee, 2016).

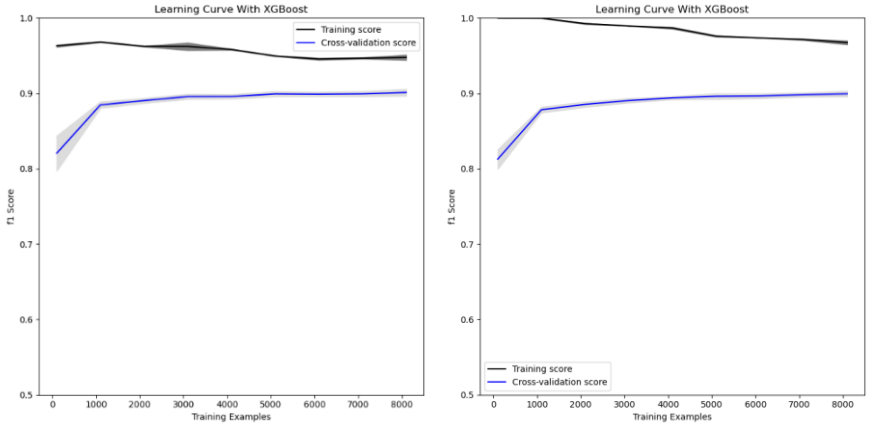

Figure 7: Salary XGBoost learning curves for the (left - a) final LC and (right - b) LC with only 2 tuned hyperparameters.

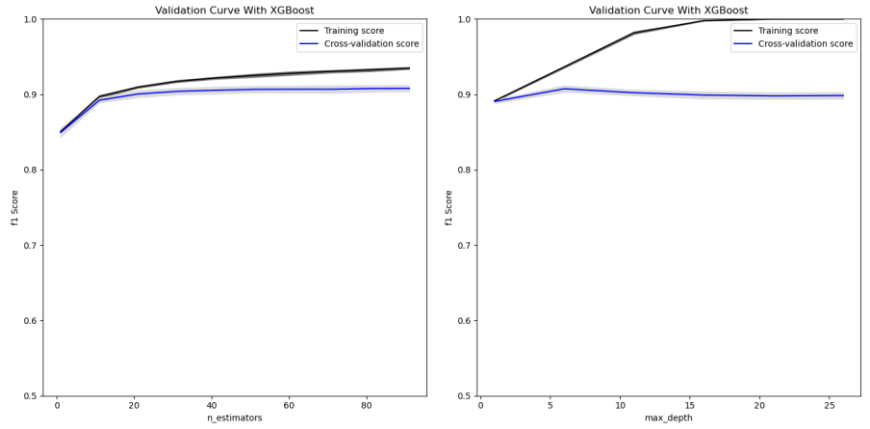

Figure 8: Validation curves for 'Salary' data tuning (left) number of estimators and (right) max depth.

4.3 *Shroom's Boost*

Shroom's Boost LC (figure 9) looks strikingly like the DT LC which makes sense. Since a single DT can achieve reliable results with a shallow tree, boosting will yield similar results as it is multiple shallow trees. This can also be seen by the fact it only needs 15 estimators where Salary needs 60. An interesting similarity is the noted dip in F1 around 300 samples. This indicates that some of the "sporadic warriors" in the data have crept in. With an altered random seed, this anomaly changes, showing edge cases do impact this data.

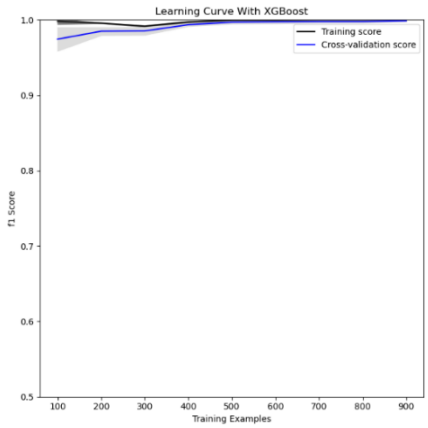

Figure 9: Final XGBoost learning curve for Shrooms data.

Something to note is the lack of variance or bias in the Shroom LCs (DT and Boost). This tells us that our training and test data are nearly identical in their distributions and behavior (low variance). It also shows that less people should be dying of mushroom poisoning since many methods can accurately divide the data into poisonous or not with the proposed features (low bias).

5 KNN

5.1 *Scaling Data*

KNN is the first algorithm requiring scaling of data since distance to a neighbor will inherently be skewed by sizing of data. Normalization was chosen (all values range between 0-1) to decrease distance skewing rather than just standardizing (mean=0 and std=1). Performance improvements might occur if alternative scaling was explored since it would alter the "distances" between points.

5.2 *Salary KNN*

5.2.1 Distance vs uniform weights

Distance and uniformed weighted KNN was explored for various values of k (Figure 10). Both CVs performed similarly. A perfect score for training in distance makes sense since it is a weighted KNN, thus testing the testing point will give the exact right answer as it will be on top of itself. It also makes sense that the training score for uniform weight trends towards no variance from the CV score for high k as one approaches extreme underfitting/generalization – which also explains the underfitting trend for the CV. Although they yield similar results, uniform weighted KNN yielded slightly better results. This is due to the sporadic data and outliers to which weighted KNN algorithms perform worse (Akben & Alkan, 2015, 150).

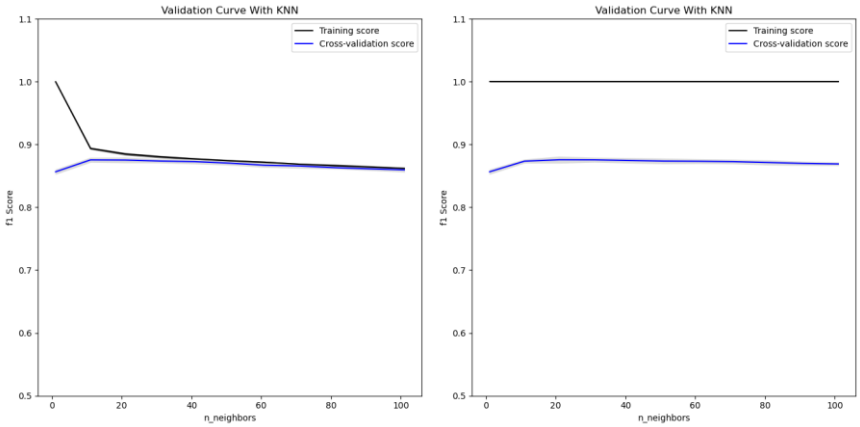

Figure 10: Salary validation curves for k values with (left) uniform weights and (right) distance weights.

5.2.2 Bias-variance and algorithm improvements

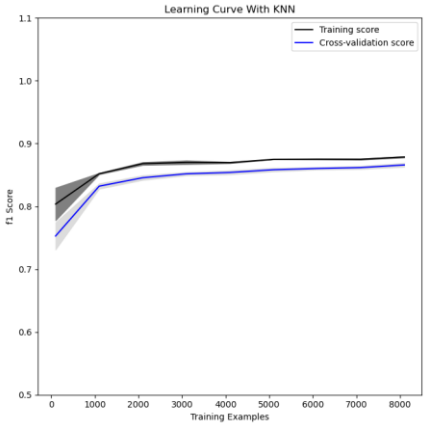

Figure 11: Salary final learning curve.

KNN algorithms are supposed to yield high variance and low bias, however, we see less variance here than any algorithm thus far (Figure 11) (Brownlee, 2016). This can partially be explained by the SMOTE oversampling which likely makes the split train-test data more similar, thus decreasing variance. However, this is also due to the poor training performance, showing we have higher bias than expected from KNN, and the algorithm has room for improvement.

Improvement could be made in three areas: alternative scaling methods (discussed previously), feature selection and extraction, and remidiating sparse data. KNN doesn't perform well with high-dimensionality data and irrelevant features since this complicates and increases distances between points that may be more similar upon feature removal (Aporras, 2018). As seen before, the features for 'Salary' are complicated and often intertwined. Redudency (dependent features) also impacts KNN through distance "skewing", and many of these features are hard to determine if they are related (e.g. race/gender and income have inherent societal biases embedded), thus feature extraction could help. Finally, sparse data and outliers can alter class boudaries and results since KNN isn't generalizing as strongly as other algorithms (like regression) (Aporras, 2018).

5.3 *Shrooms KNN*

Now back to our "I promise Shrooms are interesting" saga. Figures 12 and 13 show the extremely low bias and variance of the final model's learning curve and the validation curves. Now, how does Shrooms have such success where Salary doesn't when they both have some "whack" data, and Shrooms has *way* more features than Salary? In line with the DT results, it is likely that some of the features have very minimal impact on the "location" of the points when delineating between classes. In fact, removal of some features would likely lead to better CV results with even less data (currently need ~800 datapoints to achieve perfection even though the lines look like they are on top of each other starting sooner). This also explains why uniform and distance weighted algorithms performed similarly if points are clustered. The fact that the data didn't need rebalancing also likely improves the results.

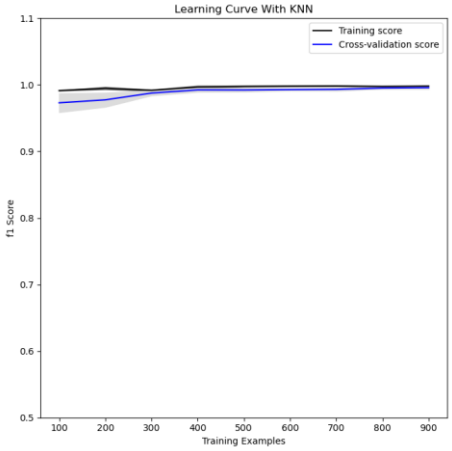

Figure 12: Shrooms KNN final learning curve.

Interestingly, the validation curves show a point of underfitting with high k, as expected, but there is never a point of overfitting. This further implies that the data is clustered so that there is a clear class boundary and similar classes are "near" each other. This further explains why Shrooms has a lower optimal k than Salary – higher k is more likely to underfit for Shrooms where high values are necessary to deal with outliers and the vast feature space in Salary.

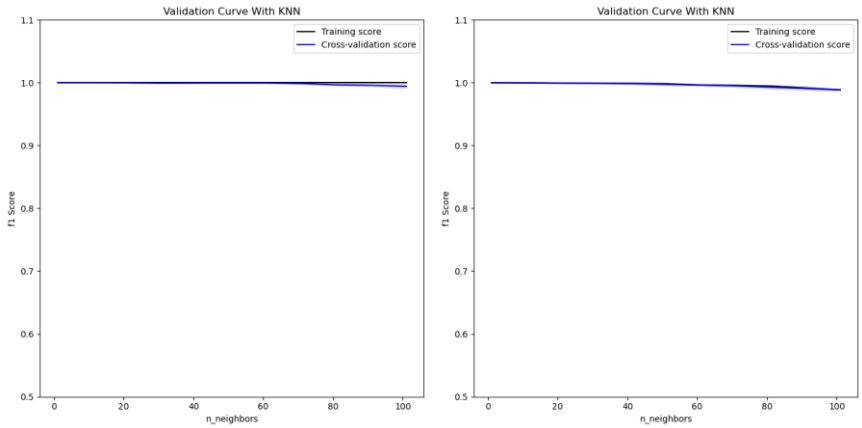

Figure 13: Shrooms validation curves for varying k with (left) distance and (right) uniform weighted KNN.

6 SVM

6.1 *Data Processing*

Data was standardized instead of normalized for building SVMs, since the sklearn documentation "assumes all features are centered around zero and variance is of the same order" (Bhandari, 2020). Even though the poly kernel was also explored, standardization was held constant. Exploring the impact of other scaling methods could improve the algorithm.

The other processing utilized was down sampling to increase the speed of the experiments. 500 positive and 500 negative random samples were utilized. This has many implications from potentially removing important edge case data to yielding higher variance. This down sample was also preformed on the already oversampled Salary data, thus this could add another layer of error.

6.2 *Kernel selection*

Since the Salary data has continually proved complicated to separate, it is unsurprisingly not linearly separable, thus a linear kernel wasn't explored, and a polynomial kernel was utilized as a more generalized linear kernel. The Gaussian radial basis function (RBF) was also selected to explore since it is commonly used and useful when there is no prior knowledge of the data (Awasthi, 2020).

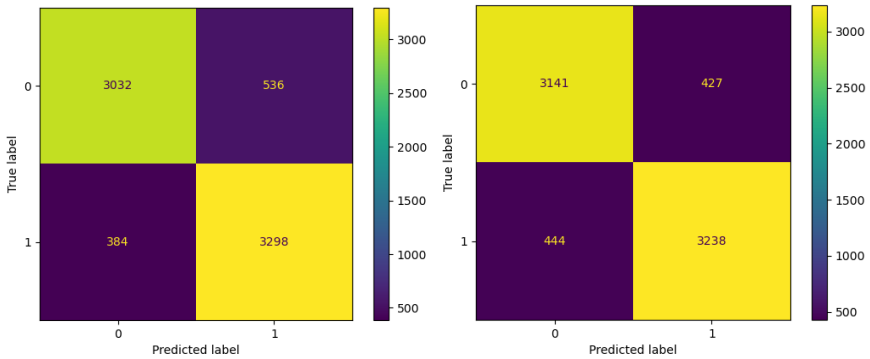

Figure 14: Confusion matrices for Salary (left) polynomial and (right) RBF kernels.

Both kernels performed similarly with Salary, so when deciding on the "better" algorithm, it is important to look at the confusion matrices (Figure 14), and ask what is more important to "get right". Poly performs better with predictions of income >$50k and RBF performs better with income <=$50k (Salary likely matters less on outcome than Shrooms which you want to minimize poisonous mushroom labeled as edible). The difference in outcome is determined by how the kernel separates the data, but it is hard to determine details of how data was labeled inaccurately in one versus the other.

Shrooms had both poly and RBF mislabel one instance of poisonous as edible. This means this instance is likely one of the outlier pieces of data.

6.3 *Tuning C*

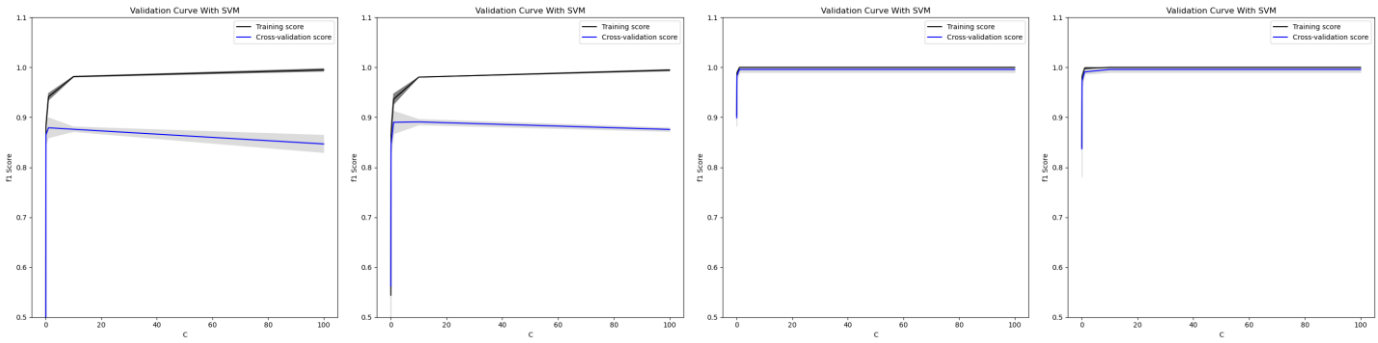

Figure 15: Validation curves for parameter C with (a) Salary Poly kernel, (b) Salary RBF kernel, (c) Shrooms Poly Kernel, and (d) Shrooms RBF kernel.

C was chosen to tune since initially a linear kernel was chosen and gamma isn't a parameter. When the data was determined to be non-linearly separable, C was retained as the hyperparameter to tune. Altering gamma could lead to some improvements in the algorithm. All kernels and data returned C=1 as the optimal C value which indicates larger margined, more generalized models perform better with the data (Yıldırım, 2020). I.e. this means accepting higher bias for a lower variance performs optimally (CV determines that allowing more misclassification performs better overall) (Starmer, 2020). This is consistent with the Salary data that we have noted is highly complex and proven by the DTs to not be easily separated and by KNN where overfitting is more of a concern than underfitting. However, too low of a C leads to underfitting as seen in figure 15. This indicates too large of a margin shifts the support vector classifier to a point of heavy misclassification, further indicating there are many points near the margin regardless of kernel.

Note Shrooms curves continue to prove there is limited variation to the data since training and CV sets perform nearly identically.

6.4 *Kernel Comparison*

Both kernels took similar times to build, but querying took nearly 3x as long for RBF than poly kernels with both datasets. This is likely due to the complexity of the RBF kernel function compared to the poly kernel and the high dimensionality of the data.

Figure 16 further shows the complexity of the RBF kernel by showing how many more samples Shrooms needs to train on than using the poly kernel. This also supports the idea that the Shrooms data is clustered, as seen with the LC and low k in KNN. Since RBF is essentially looking at weighting by close points in infinite dimensions, the "clusters" need enough similar data points to accurately draw the SV classifier where the poly function just needs enough to draw a multi-dimension plane between the points.

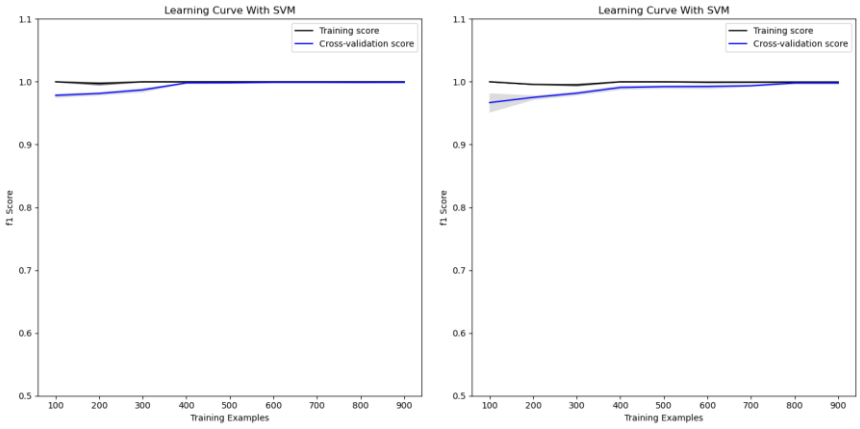

Figure 16: Shrooms learning curves for (left) poly and (right) RBF kernels.

Both kernels display a level of bias for Salary, as expected, based on the C value bias-variance tradeoff discussed earlier (Figure 17). RBF has less variance than poly, which is expected, due to the complexity of the Salary data – RBF would more consistently classify complex data than poly. However, RBF shows a tendency for overfitting with lower amounts of data than poly with regard to the training data. Since RBF is more similar to KNN ideals, the margins drawn on a smaller amount of data is less likely to generalize to CV

data as the “neighbors” will vary more with lower amounts of data. This is also validated by the similarly mentioned trend in Shrooms (needing more data to achieve desired results).

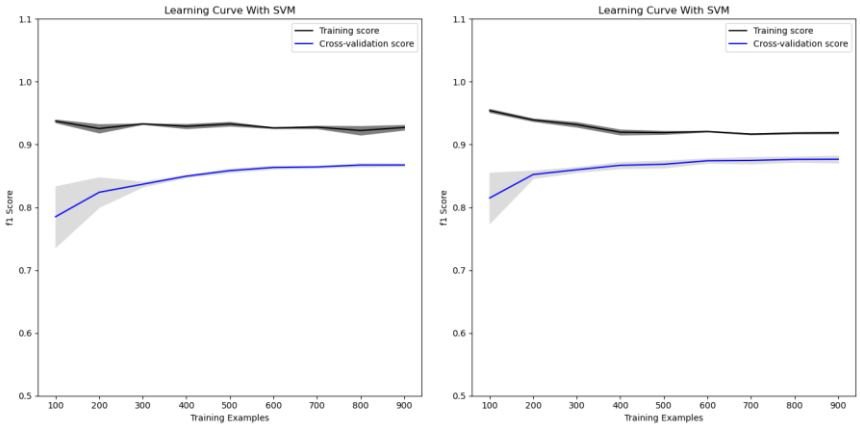

Figure 17: Salary learning curves for (left) poly and (right) RBF kernels.

As mentioned, the best opportunities for improvement lay in tuning gamma and reducing the dimensionality (features) of the space.

7 ANN

7.1 *Adding Layers*

One, three, and five hidden layers were explored holding learning rate constant at 0.01. Figure 18 reveals that additional layers lead to higher likelihood of over fitting at higher epochs since more layers = more learning of the training data = worse performance on CV. Because of this, one layer was chosen for the final model in both datasets.

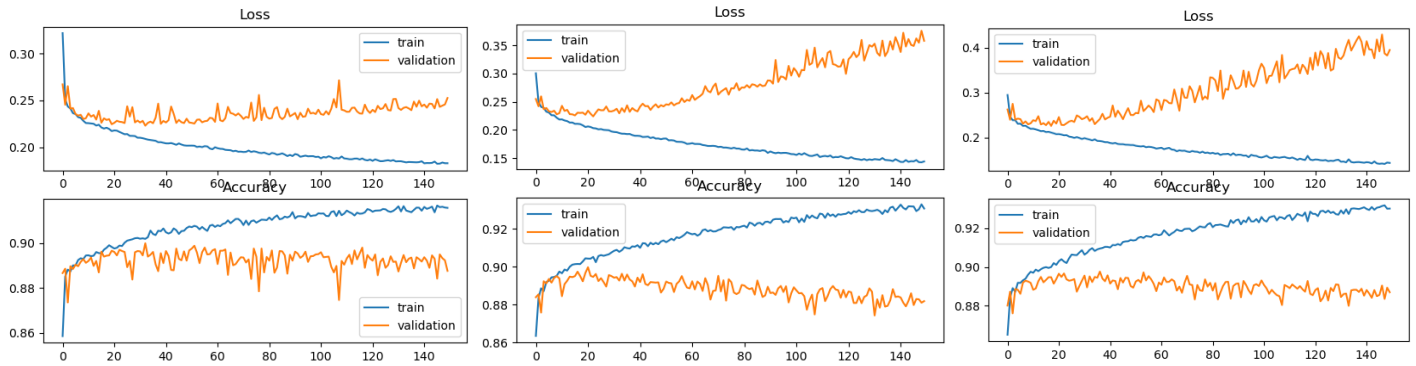

Figure 18: Loss and accuracy curves for Salary with (left) 1, (middle) 3, and (right) 5 identical layers.

7.2 *Altering Learning Rate*

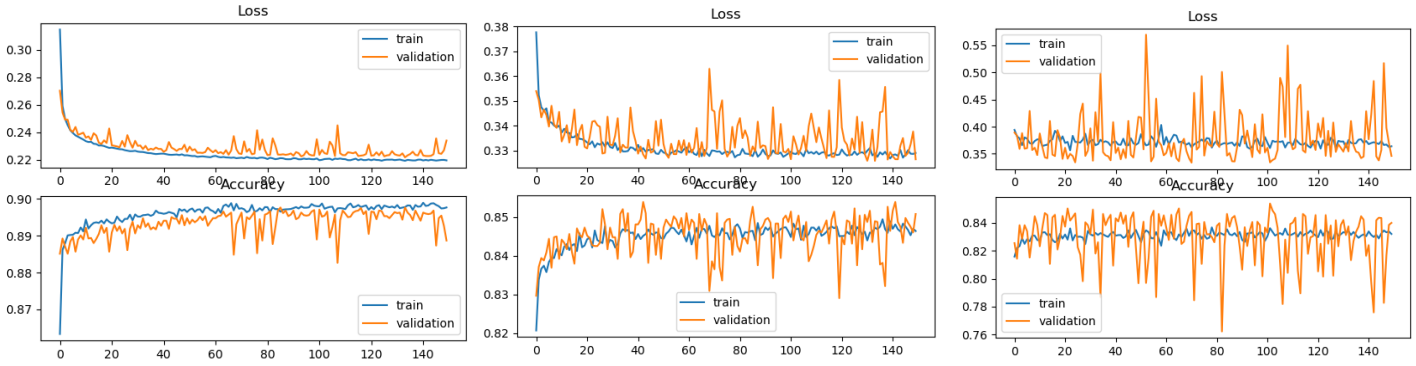

Figure 19: Salary loss curves for (left) 0.05, (middle) 0.1, and (right) 0.5 learning rates.

Using the knowledge of one layer being optimal, learning rate was then varied between 0.05, 0.1, and 0.5 since it’s said “if you have time to only tune one-parameter, tune the learning rate” (Karani, 2020). Figure 19 shows how increasing learning rate progresses from

taking too long to learn (need many epochs) to being so large that the minimum loss function is never found (bouncing around a point vs decreasing). The trade off then becomes time vs finding the minimum loss function.

Looking at the Shrooms loss for various learning rates in figure 20, we can see the same tradeoff for learning time, but there doesn't seem to be the overfitting tendencies, further supporting that there is clear separability in the data. The curves are also much smoother than Salary's, continuing to show us that Shrooms data has minimal variance and is easy to predict.

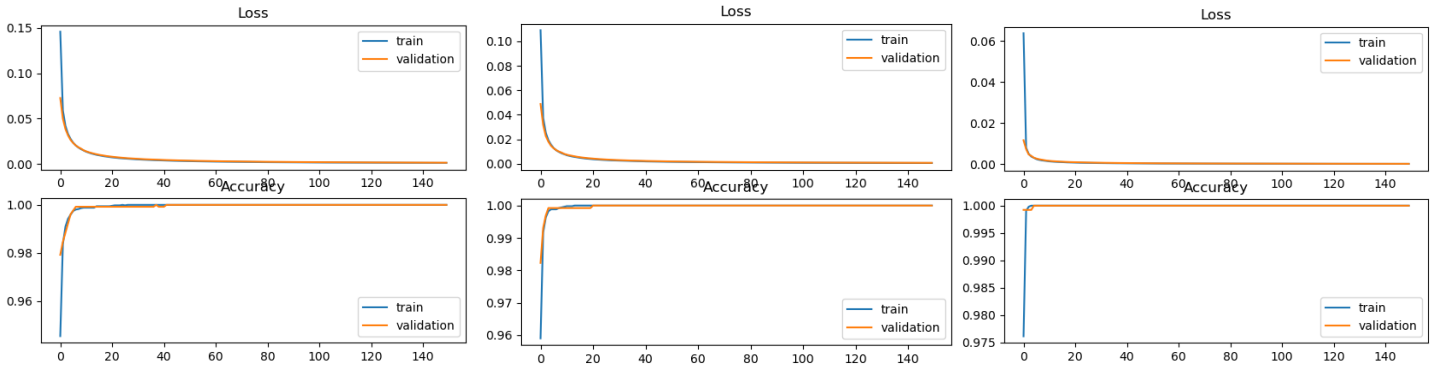

Figure 20: Shroom loss curves for (left) 0.05, (middle) 0.1, and (right) 0.5 learning rates.

7.3 *Final Models and Improvements*

Final LCs, showing training vs test data, (Figure 21) show that the ANNs could be succesfully trained in about 20 epochs for Salary before overfitting occurs. The final model chose a slightly higher learning rate to speed training, but it's not so high that a minimum is never found, thus overfitting must be accounted for. The sporadic nature of the testing data would be smoothed out by adding more data, but also indicates the complexity of the Salary data compared to the Shrooms data. Shrooms, yet again, shows us that the train and test data has little variance and that this data can be very accurately classified with very few epochs.

There is definite opportunity for improving the Salary data through feature selection and the fine tuning of many of the unexplored hyperparameters like momentum and neural density.

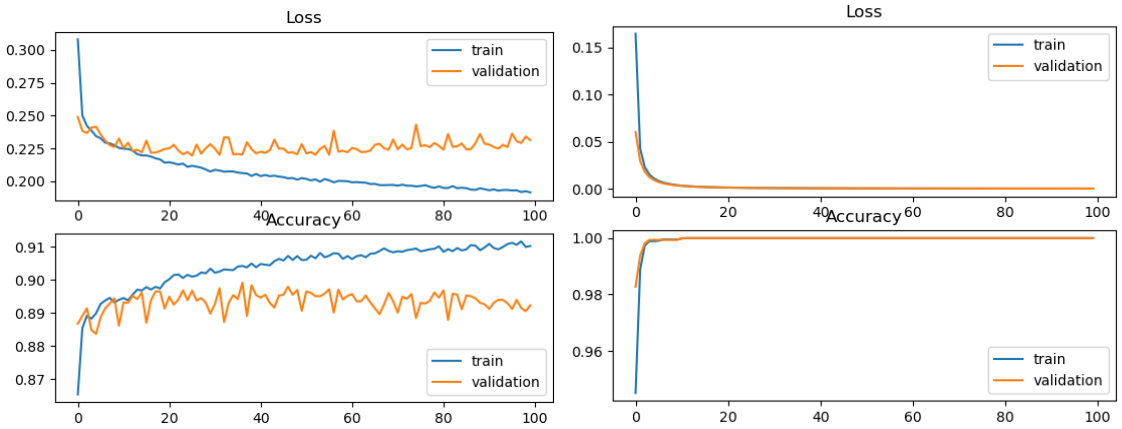

Figure 21: Final learning curves for (left) Salary and (right) Shrooms ANNs with train and test data.

8 References

1. Adult. (1996). UCI Machine Learning Repository.
2. Akben, S. B., & Alkan, A. (2015, November). Density Weighted K-Nearest Neighbors Algorithm for Outliers in the Training Set Are So Close to the Test Element. *Journal of Electrical Engineering*, *3*(3), 150-161. 10.17265/2328-2223/2015.03.006
3. Aporras. (2018, July 18). *10 Reasons for loving Nearest Neighbors algorithm*. Quantdare. Retrieved September 21, 2022, from https://quantdare.com/10-reasons-for-loving-nearest-neighbors-algorithm/
4. Awasthi, S. (2020, December 17). *Seven Most Popular SVM Kernels*. Dataaspirant. Retrieved September 23, 2022, from https://dataaspirant.com/svm-kernels/#t-1608054630726
5. Aznar, P. (2020, December 2). *Decision Trees: Gini vs Entropy*. Quantdare. Retrieved September 21, 2022, from https://quantdare.com/decision-trees-gini-vs-entropy/

6. Bengio, Y., Delalleau, O., & Le Roux, N. (2005, March 2). The Curse of Dimensionality for Local Kernel Machines. *Departement ´ d'Informatique et Recherche Operationnelle*, (1258), 1. http://www.iro.umontreal.ca/~lisa/pointeurs/tr1258.pdf
7. Bhandari, A. (2020, April 3). *Feature Scaling | Standardization Vs Normalization*. Analytics Vidhya. Retrieved September 21, 2022, from https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
8. Brownlee, J. (2016, March 18). *Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning*. Machine Learning Mastery. Retrieved September 21, 2022, from https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/
9. Brownlee, J. (2016, September 7). *How to Tune the Number and Size of Decision Trees with XGBoost in Python*. Machine Learning Mastery. Retrieved September 21, 2022, from https://machinelearningmastery.com/tune-number-size-decision-trees-xgboost-python/
10. Karani, D. (2020, March 15). *Experiments on Hyperparameter tuning in deep learning — Rules to follow*. Towards Data Science. Retrieved September 23, 2022, from https://towardsdatascience.com/experiments-on-hyperparameter-tuning-in-deep-learning-rules-to-follow-efe6a5bb60af
11. Kavish. (2022, May 30). *Handling Imbalanced Data with Imbalance-Learn in Python*. Analytics Vidhya. Retrieved September 21, 2022, from https://www.analyticsvidhya.com/blog/2022/05/handling-imbalanced-data-with-imbalance-learn-in-python/
12. Mushroom. (1987). UCI Machine Learning Repository.
13. Nikulski, J. (2020, March 16). *The Ultimate Guide to AdaBoost, random forests and XGBoost*. Towards Data Science. Retrieved September 21, 2022, from https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f
14. Starmer, J. (Writer). (2020). Support Vector Machines Part 1 (of 3): Main Ideas!!! [TV series episode]. In *StatQuest*. https://www.youtube.com/watch?v=efR1C6CvhmE&list=PLblh5JKOoLUL3IJ4-yor0HzkqDQ3JmJkc&index=2&t=1s
15. 2U, inc. (2022, April). *What Is Undersampling?* Master's in Data Science. Retrieved September 21, 2022, from https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/
16. Yıldırım, S. (2020, October 6). *SVM Hyperparameters Explained with Visualizations | by Soner Yıldırım*. Towards Data Science. Retrieved September 23, 2022, from https://towardsdatascience.com/svm-hyperparameters-explained-with-visualizations-143e48cb701b