CS 7641: Machine Learning
Assignment 1: Supervised Learning

Mahmoud Shihab
mshihab6@gatech.edu

Supervised Learning on the Adult Census Income / Wine Quality Data Sets

Datasets:

Adult Census Income[1]: An unbalanced dataset with the aim of predicting if someone income exceeds $50K/year. A dataset like this can be useful to see what kinds of factors contribute to a family's/households income

Why is this dataset interesting? While being a binary classification, the dataset consists of 30,162 entries of mainly categorical data, making it a sufficiently large dataset. However, 75% of the data is only one class [<=$50k]. This makes the task of classifying those with more than $50k a much harder task.

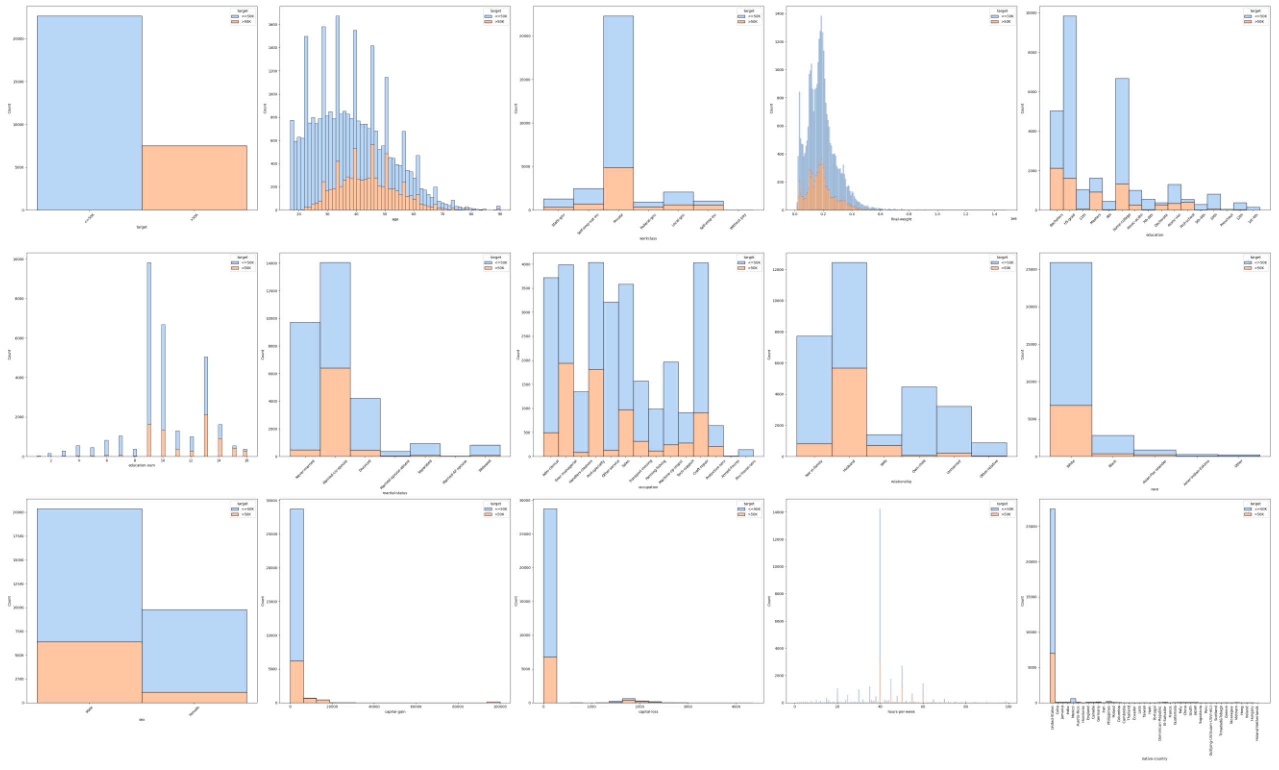


Figure 1: Adult Dataset Histograms

Wine Quality[2]: An unbalanced dataset with the aim of predicting the quality of a particular type of wine on a scale *(originally from 3 – 8 but change to 0 – 5 in the analysis).*

Why is this dataset interesting? This dataset can be taken in either a classification or regression direction for predicting the quality of a particular wine. The dataset the opposite of the adult dataset in that it consists of 1143 entries which are mainly numeric, which is pretty small. The quality target can either be looked at a set of 6 classes or a range of integers that go from 0 to 5. In either case, most of the "quality" data points seems to exist around 2-3 and very few at the ends, which can make predicting challenging. We will focus on the classification setting of this problem, but it would be interesting to see the differences between classification and regression.

[1] Becker,Barry and Kohavi,Ronny. (1996). Adult. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20.
[2] https://www.kaggle.com/datasets/yasserh/wine-quality-dataset

Mahmoud Shihab
mshihab6@gatech.edu

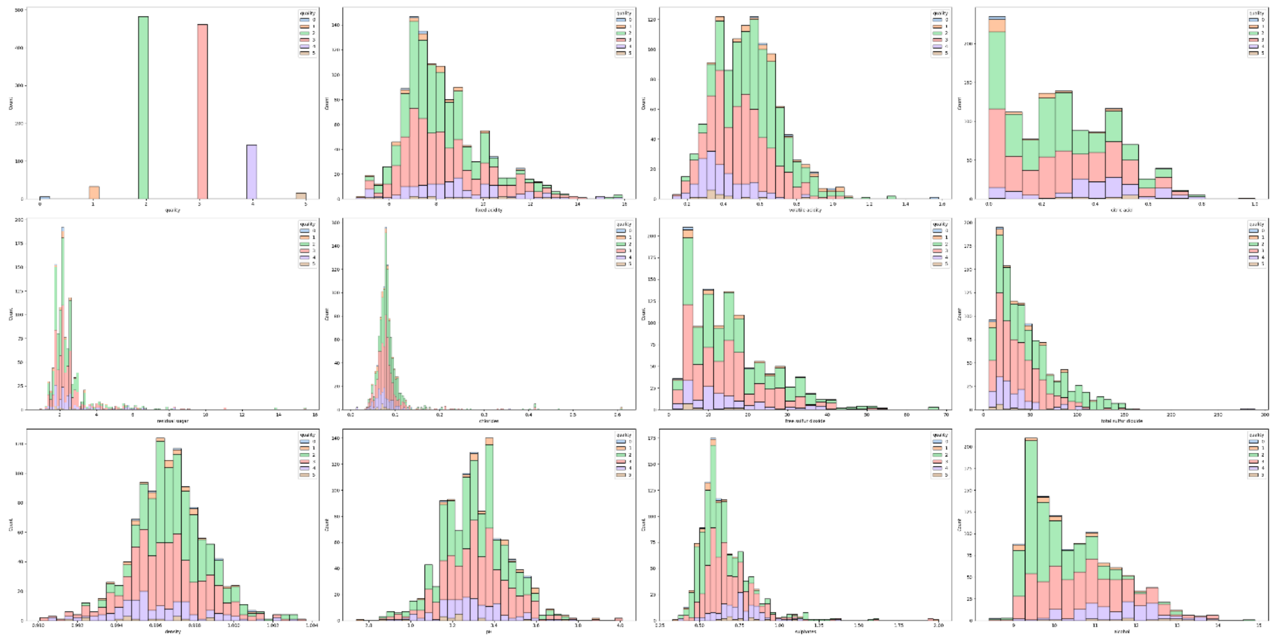


Figure 2: Wine Data Histograms

Methodology:

This assessment will be done in Python using:

- Scikit-Learn (version 1.3.0)
- Pandas (version 2.0.3)
- NumPy (version 1.25.2)

For each dataset:

- NULL and duplicated values will be searched for and removed using Pandas and NumPy.
- Dataset will be scaled (for numeric data) using the `StandardScaler` function and encoded (for categorical data) using the `LabelEncoder` function
- Data will be oversampled (to avoid data loss) using the `imblearn` library
- Model will be trained using the `train_test_split` with a `test_size` of 30%
- Model selection will be done through `GridSearchCV` with a `cv` = 10
- The scoring metric will be set to `Accuracy`.
- All random processes will be set with a `random_state` = 42

There are five algorithms being tested:

- K-Nearest Neighbors: `from sklearn.neighbors import KNeighborsClassifier`
  - `Alias: KNN`
  - `n_neighbors = [3, 5, 7, 9]`
  - `p = [1, 2]` `(Manhattan vs Euclidian Distance)`
- Decision Tree: `from sklearn.tree import DecisionTreeClassifier`
  - `Alias: DTC`
  - `criterion = [‘gini’, ‘entropy’, ‘log_loss’]`
  - `max_depth = [None, 2, 3, 4, 5, 6, 7, 8, 9]`
  - `min_samples_split = [2, 3, 4, 5]`
  - `min_samples_leaf = [1, 2, 3, 4, 5]`

- Ada Boosted Trees[3]: `from sklearn.ensemble import AdaBoostClassifier`
  - `Alias: ABC`
  - `n_estimators = [50, 100, 150, 300]`
- Support Vector Machines: `from sklearn.svm import SVC`
  - `Alias: SCV`
  - `kernel = ['linear', 'poly', 'rbf', 'sigmoid'`
  - `degree = [3, 4, 5] (Only used when kernel is 'poly')`
- Neural Networks: `from sklearn.neural_network import MLPClassifier`
  - `Alias: NN`
  - `hidden_layer_size = [(5,),(5,5),(5,5,5), (10,),(10,10),(10,10,10), (15,),(15,15),(15,15,15), (20,),(20,20),(20,20,20), (30,),(30,30),(30,30,30), (50,),(50,50),(50,50,50), (100,),(100,100),(100,100,100)]`
  - `early_stopping = [True, False]`

A note on SMOTE:

SMOTE stands for Synthetic Minority Over-sampling Technique. It works in 4 steps (quoted[4]):

1. Choose a minority class as the input vector.
2. Find its k nearest neighbors (k_neighbors is specified as an argument in the SMOTE() function).
3. Choose one of these neighbors and place a synthetic point anywhere on the line joining the point under consideration and its chosen neighbor.
4. Repeat the steps until the data is balanced.

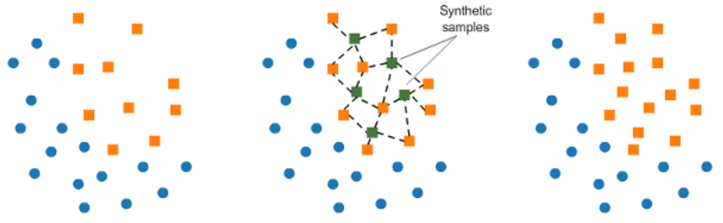


Figure 3: A representation on how SMOTE works

The reason this method was chosen since the Wine dataset has classes that have datapoints in the single or double digits. Having the same datapoints be repeated 2,000 times felt like it may lead to some overfitting when it came to those particular classes.

Algorithm Implementation on Adult Census Dataset:

No Oversampling

All the algorithms seemed to perform roughly the same (Accuracy ≈ 85% ± 2%). However, due to the imbalance of the data, it was very strong in predicting those who have <=$50k annual salaries.

[3] *It is worth noting that the default estimator for AdaBoostClassifier is a DecisionTreeClassifier with max_depth=1. AdaBoostClassifier can use other types of estimators other than DecisionTreeClassifier*

[4] guest_blog. (2023, April 26). 10 Techniques to Solve Imbalanced Classes in Machine Learning (Updated 2023). Retrieved from Analytcs Vidhya:
https://www.analytcsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-i m balance-i n-machi ne- p

Mahmoud Shihab
mshihab6@gatech.edu

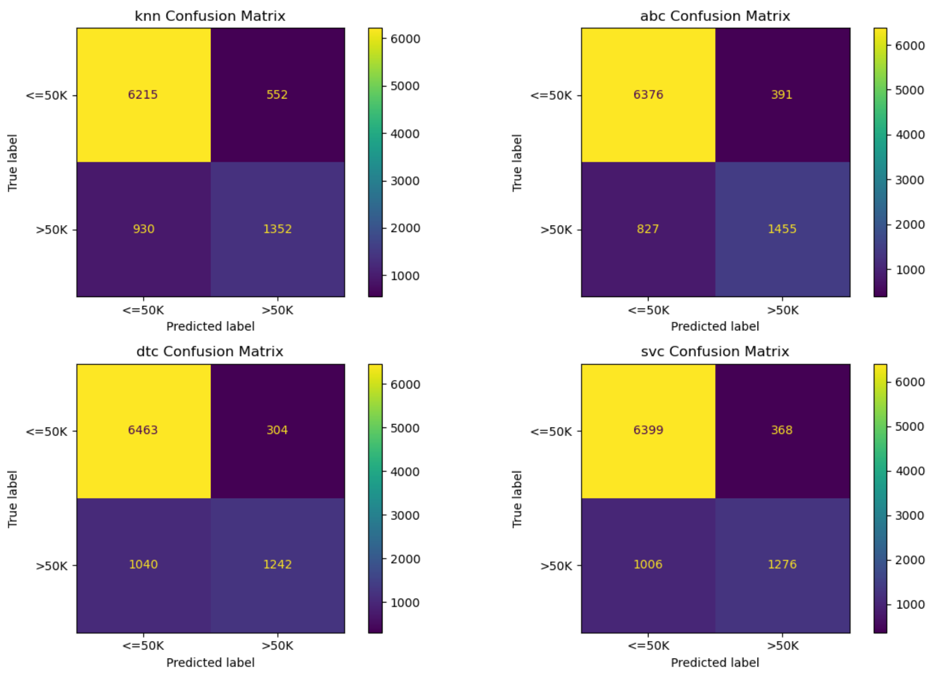


Figure 4: Confusion Matrix for 4 out of the 5 algorithms on the Adult Dataset using no Oversampling

You'll notice that for those with >$50k, it's pretty much left to chance. As stated in the Dataset portion of the report, the ratio of <=$50k:>$50k is 3:1, which means that the model doesn't have enough data on the >$50k class, even though there are roughly 7,000+ points.

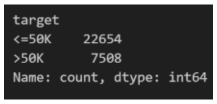

```
target
<=50K    22654
>50K      7508
Name: count, dtype: int64
```

Figure 5: Adult Target Class Counts

Let's try oversampling.

SMOTE Oversampling

All the algorithms again seemed to perform roughly the same (Accuracy ≈ 85% ± 2%), but this time, with more datapoints in the <$50k class. The top performers are KNN, AdaBoosClassifier, and Neural Networks with an accuracy of 87%

The algorithms, after trained and GridSearch-ed on the SMOTE dataset, were tested using both the original data and the SMOTE data. The confusion matrix below show that the algorithms seem to be performing better in predicting the under-represented class.

Mahmoud Shihab
mshihab6@gatech.edu

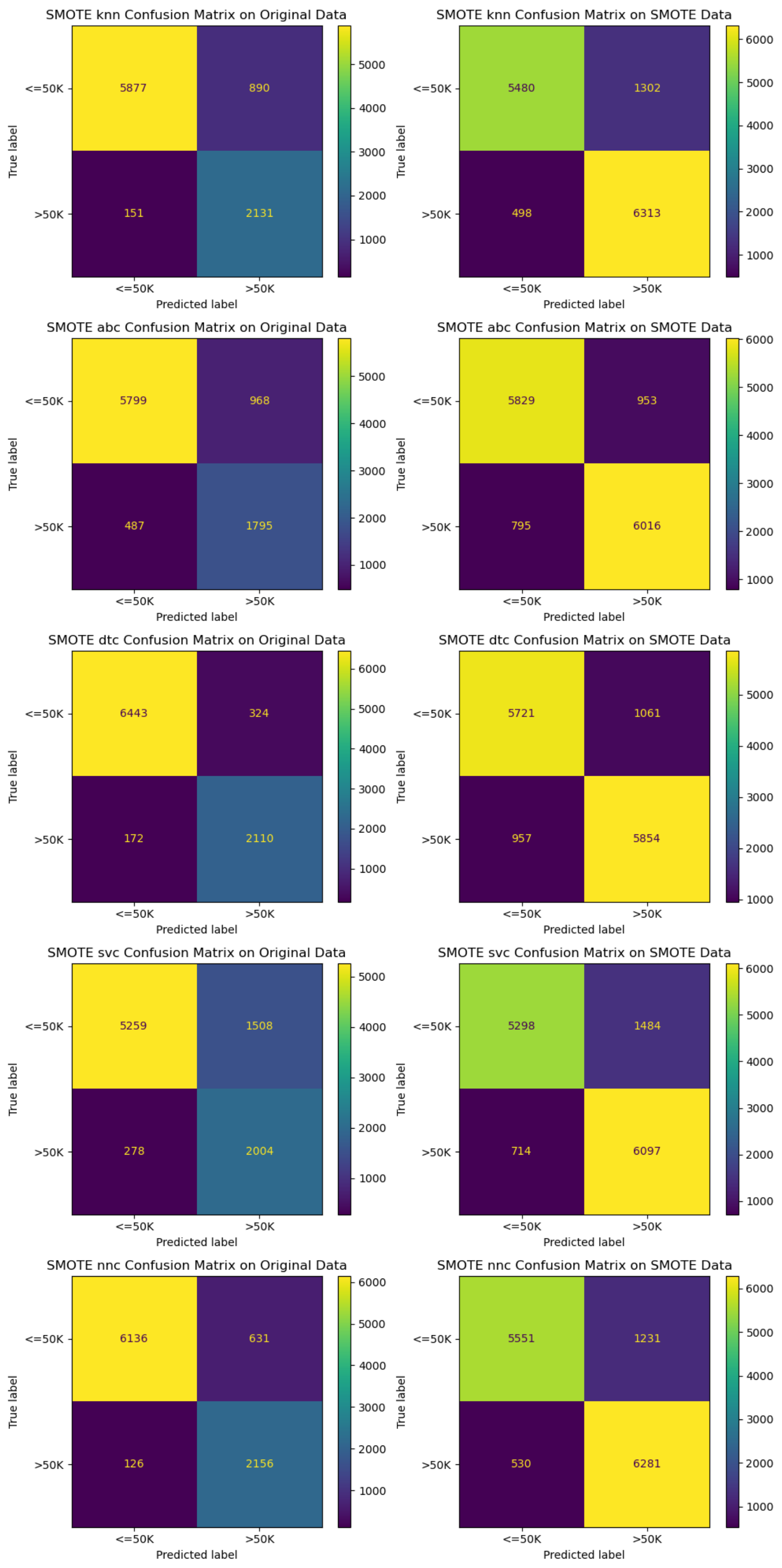
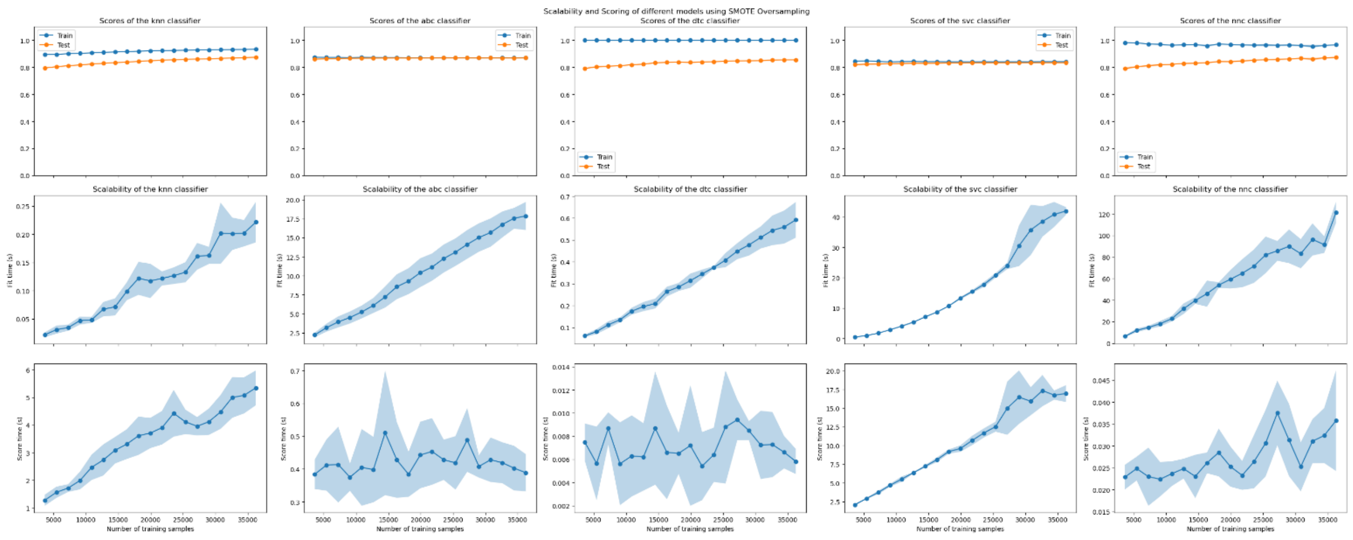


Figure 6: Confusion Matrix for Adult Dataset using SMOTE Oversampling on Original and SMOTE-ed Versions

Mahmoud Shihab
mshihab6@gatech.edu



Looking at the learning curves, it seems like that each model (excluding ABC and SVC) benefits from the plethora of data at its disposal, getting closer to its true accuracy as more data is presented in both the training and testing phases. Neural networks seem to exhibit a much larger gap in their learning curve than the others. However, KNN and DTC seem to take very little to no time when training, probably due to the simple nature of their algorithms. ABC is a bit slower given that it a bunch of weak and low depth DTCs, but is still much faster than SCV, which can take close to a full minute to train and half a minute to test, or Neural Networks that could take around 3 minutes to train.

Figure 7: Learning Curves for Adult Census Dataset

model	3624	5436	7249	9061	10873	12686	14498	16310	18123	19935	21747	23559	25372	27184	28996	30809	32621	34433	36246	optimal_train_size	test_scores
abc	0.874	0.873	0.873	0.871	0.873	0.872	0.872	0.871	0.870	0.871	0.870	0.871	0.871	0.871	0.870	0.870	0.871	0.870	0.871	3624	0.860
dtc	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	36246	0.855
knn	0.894	0.896	0.900	0.903	0.906	0.910	0.913	0.916	0.919	0.921	0.923	0.925	0.926	0.927	0.928	0.930	0.931	0.932	0.934	36246	0.874
nnc	0.982	0.980	0.972	0.969	0.963	0.967	0.968	0.959	0.972	0.968	0.966	0.963	0.964	0.963	0.964	0.961	0.955	0.961	0.966	3624	0.791
svc	0.845	0.847	0.843	0.840	0.842	0.843	0.842	0.842	0.841	0.841	0.840	0.841	0.841	0.841	0.841	0.841	0.842	0.842	0.842	5436	0.823

Table 1: Train Scoring Data for Adult Census Dataset

model	3624	5436	7249	9061	10873	12686	14498	16310	18123	19935	21747	23559	25372	27184	28996	30809	32621	34433	36246	optimal_train_size	train_scores
abc	0.860	0.863	0.864	0.866	0.867	0.868	0.868	0.869	0.869	0.869	0.870	0.870	0.870	0.870	0.869	0.869	0.869	0.869	0.869	3624	0.874
dtc	0.792	0.803	0.808	0.812	0.819	0.823	0.833	0.837	0.838	0.836	0.839	0.841	0.845	0.847	0.848	0.849	0.853	0.855	0.855	36246	1.000
knn	0.794	0.804	0.812	0.817	0.824	0.829	0.834	0.838	0.844	0.848	0.852	0.856	0.858	0.861	0.863	0.865	0.868	0.871	0.874	36246	0.934
nnc	0.791	0.804	0.812	0.819	0.821	0.828	0.830	0.834	0.843	0.842	0.846	0.852	0.856	0.858	0.861	0.866	0.861	0.869	0.873	3624	0.982
svc	0.819	0.823	0.824	0.827	0.828	0.830	0.829	0.830	0.832	0.832	0.833	0.833	0.834	0.834	0.833	0.834	0.834	0.834	0.834	5436	0.847

Table 2: Test Scoring Data for Adult Census Dataset

model	3624	5436	7249	9061	10873	12686	14498	16310	18123	19935	21747	23559	25372	27184	28996	30809	32621	34433	36246	optimal_train_size	score_times
abc	2.2245	3.1780	3.9327	4.5152	5.2385	6.1020	7.2084	8.5600	9.2937	10.4064	11.1382	12.2656	13.1052	14.1221	15.0301	15.6819	16.7475	17.5578	17.8635	3624	0.3842
dtc	0.0614	0.0810	0.1121	0.1349	0.1743	0.1955	0.2091	0.2641	0.2851	0.3138	0.3444	0.3743	0.4078	0.4487	0.4774	0.5114	0.5441	0.5590	0.5924	36246	0.0058
knn	0.0219	0.0310	0.0346	0.0470	0.0484	0.0672	0.0713	0.0990	0.1218	0.1174	0.1216	0.1266	0.1327	0.1609	0.1625	0.2017	0.2008	0.2019	0.2214	36246	5.3422
nnc	6.3613	11.8988	14.6264	18.0748	22.7815	31.9849	39.6066	46.0142	53.9467	59.3680	64.8883	71.4620	81.9772	85.7256	89.9181	83.0881	96.2936	91.5016	121.3094	3624	0.0229
svc	0.4486	0.9583	1.7504	2.8565	4.0661	5.3730	7.0959	8.6834	10.6555	13.2547	15.4189	17.7547	20.7337	23.8793	30.4863	35.7037	38.5136	40.7940	41.9252	5436	2.9294

Table 3: Fit Times for Adult Dataset

Looking at the underlying data of the learning curves, (unsurprisingly) KNN benefits greatly from having the full dataset at its disposal, getting more accurate and not slower in any regards.

ABC doesn't seem to benefit from the extra data as it has a stable training and testing scores, but has a rapidly increasing fit time. It's score time however seems relatively flat, which makes sense given that it's a series of decision trees, which are very fast in predicting.

Similar story for the SVC. With a large fit AND score time for little to know benefit in terms of actual training and testing scoring, the SVC seems to prefer smaller portions of data[5]. Seeing as that SVC is trying to iteratively find the best

[5] Saini, A. (2023, July 7). Guide on Support Vector Machine (SVM) Algorithm. Retrieved from Analytcs Vidhya: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Mahmoud Shihab
mshihab6@gatech.edu

hyperplane by maximizing the margin and minimizing the errors, it would take time since dealing with multiple dimensions.

Neural Networks seem to take the most time, since it needs to adjust the weights during the training, and having more and larger hidden layers will increase the complexity of the model. Surprisingly, this model doesn't really benefit from a large hidden layer count. The accuracy of the model for most layer sizes seems pretty stable (except when the layer count is or exceeds 50). At which point it may be getting close to overfitting. The model selected by the GridSearch is MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=2000, random_state=42) since it has the highest training accuracy, but that is misleading as shown by the validation curve below. Due to the limitations of the Scikit-Learn variant of Neural Networks, parameters like drop out & epochs couldn't be tested, although I suspect they won't make much of a difference given that this simple model is already performing at a constant accuracy.

DTC may be overfitting (despite what the testing scores and confusion matrices say), having a training score of 1 across the board. After further investigation, the best tree selected was DecisionTreeClassifier(criterion='entropy', random_state=42), which means that it took the default options for DTC. The maximum depth by default is None, which means it tries to fit the tree with unlimited depth, or in this case, a depth of 52. The model is probably overfitting, considering the testing scores are closer to 85%. It seems that a level of pruning could have beneficial, but since GridSearch tries the entire parameter space and find the model that provides the best accuracy, it chose the model that overfitted. *(See validation curve below)*

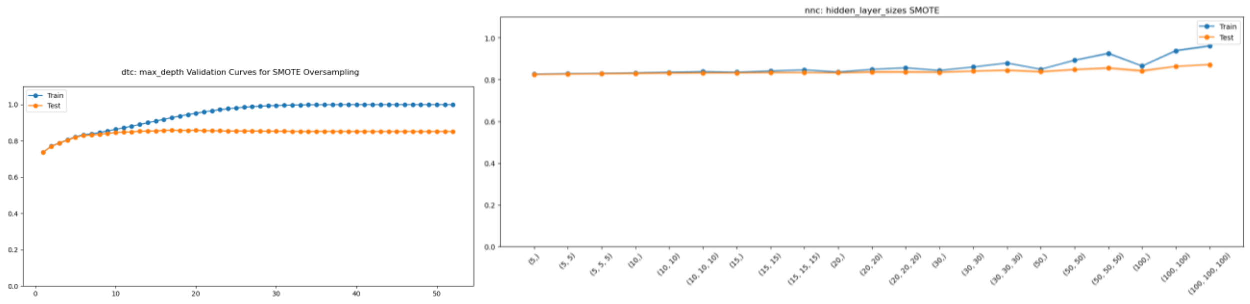


Figure 8: DTC Max Depth and Neural Network Layer Sizes Validation Curves

Algorithm Implementation on Wine Quality Dataset:

With this dataset, since the full data set is only a little more than 1000 entries, and unbalanced at that, this analysis will move forward with the SMOTE analysis. Also, since this dataset is small and can experiments can be tested quickly be tested, the use of Random Oversampling (R.OS) will be presented alongside the SMOTE analysis.

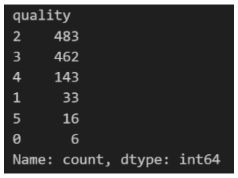

```
quality
2    483
3    462
4    143
1     33
5     16
0      6
Name: count, dtype: int64
```

Figure 9: Wine Quality Class Counts

There **seems** to be a better performance using the R.OS where the best models are DTC and KNN with 84% and 81% accuracy respectively, vs the 76% and 71% accuracy. However, when looking at the classification reports of these models, specifically the recall for the severely underrepresented quality classes (0, 1, and 5), the R.OS models tend to have a recall scores $\left(\frac{True\ Positive}{True\ Positive + False\ Positives}\right)$ equal to 1, which suggests that there is something going on. The learning curves also seem to favor the R.OS technique despite my initial fear of having R.OS lead to overfitting.

Mahmoud Shihab
mshihab6@gatech.edu

It is worth mentioning that the overall performance (not just accuracy) seems to be a bit lower than the performance of the models in the Adult Dataset. I suspect that the very low amount of data and potential outliers may be contributing to this poor performance.

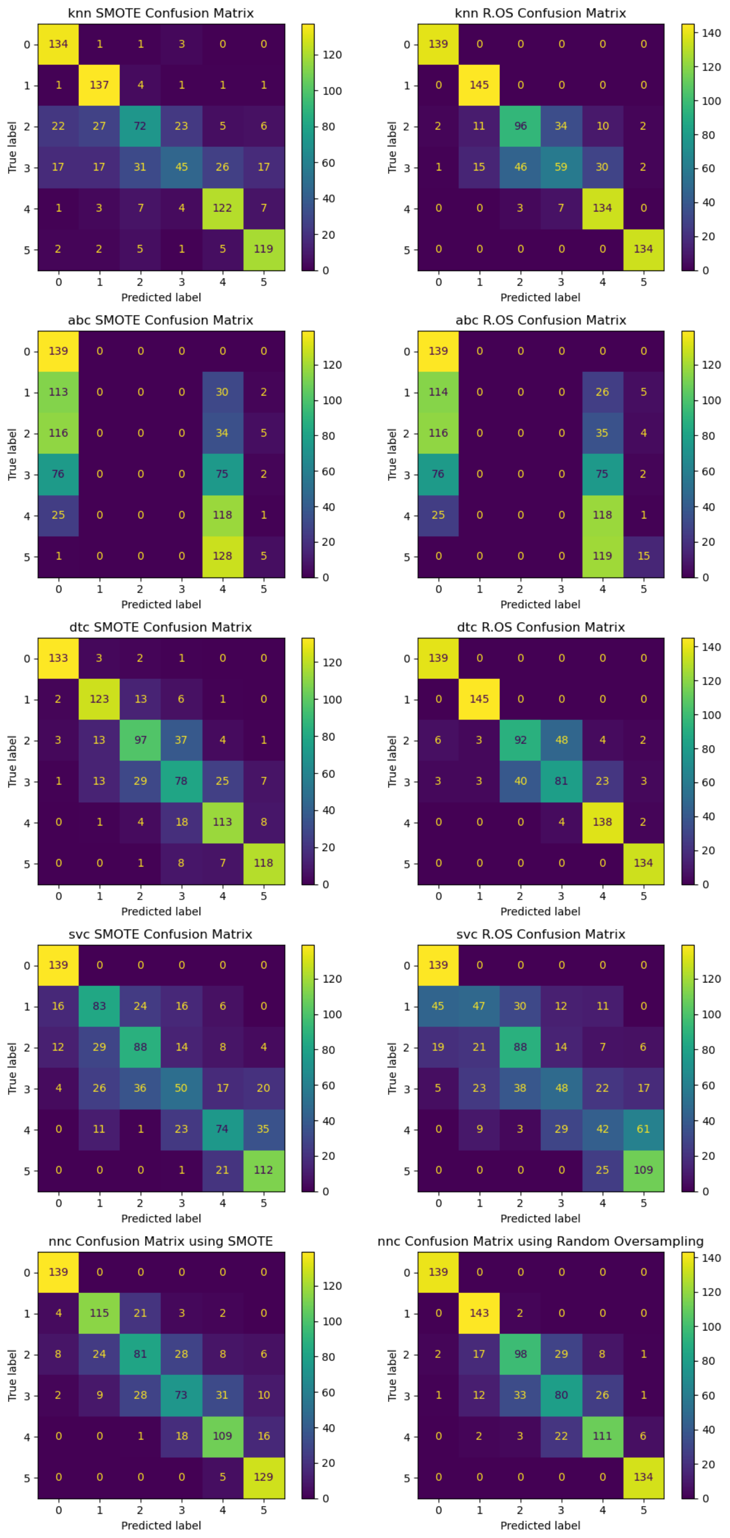


Figure 10: Confusion Matrix for Wine Dataset using SMOTE (Left) and Random Oversampling (Right)

Mahmoud Shihab
mshihab6@gatech.edu

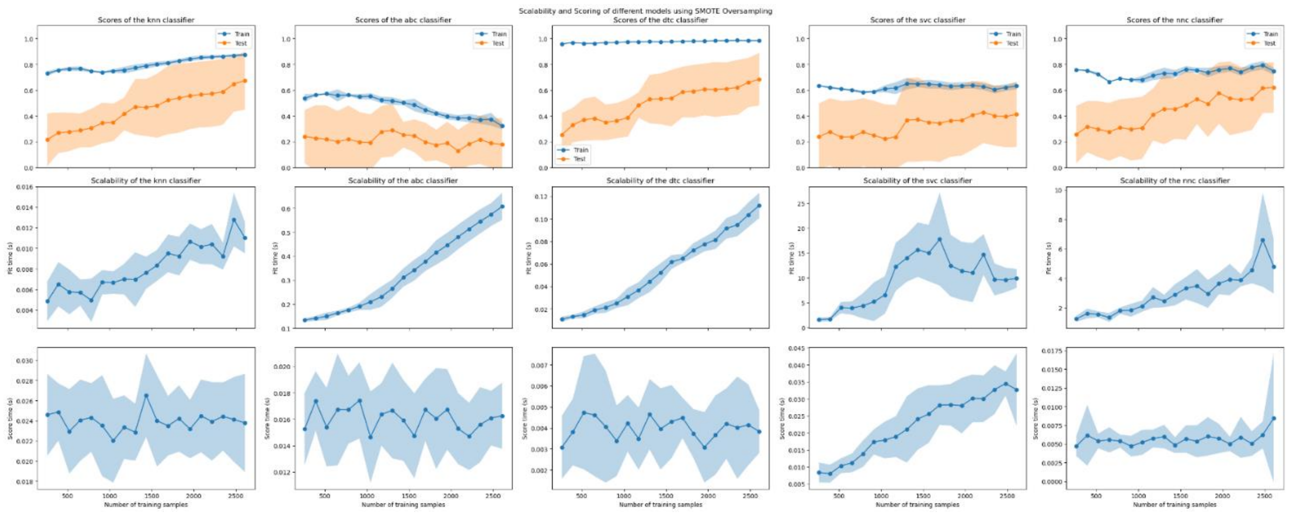


Figure 11: Learning Curves for the Wine Dataset using SMOTE

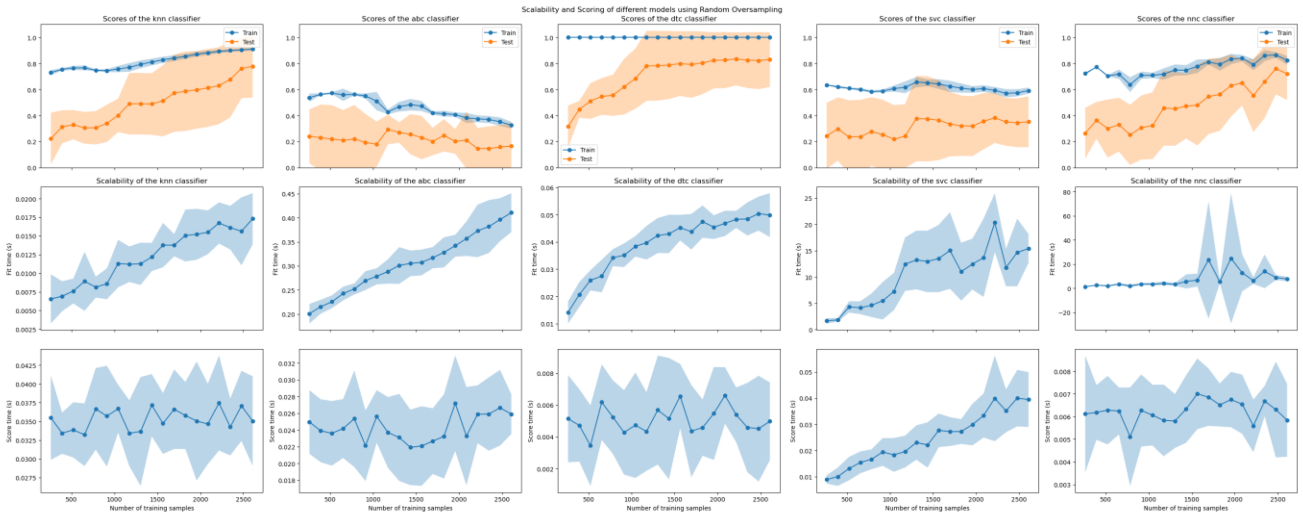


Figure 12: Learning Curves for the Wine Dataset using R.OS

Train and Score time are very comparable to Adult Dataset regardless of which up sampling method is chosen. KNN and DTC are fastest, ABC is a bit slower, and SVC + NN are the slowest.

Also, as before, DTC for both SMOTE and R.OS also seem to be suspicious close to 1. When looked at again, we see that the depths for both are set to None, or in this case, 16 & 17 respectively. R.OS however seems to have a much better score than SMOTE does. Pruning may be a beneficial action to take.

Unlike before, NN seems to be desperate to for data as the testing curve seems to be moving higher and higher towards the true accuracy. Also, the different hidden layer sizes seem to affect the accuracy more than before, especially in the R.OS case, unlike in the Adult Dataset. I suspect that since the data is small, it needs more connections to identify the pattern in the dataset for it to make an accurate prediction.

Mahmoud Shihab
mshihab6@gatech.edu

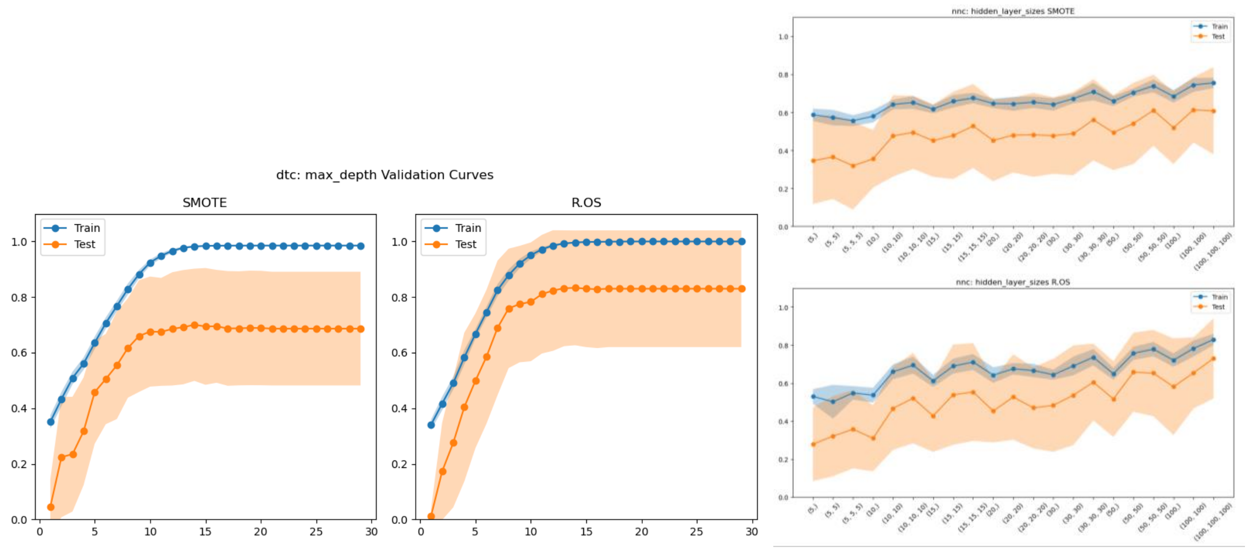


Figure 13: DTC Max Depth and Neural Network Layer Sizes Validation Curves

Unlike the previous dataset, the ABC and SVC both seem to struggle, with a very poor score. This could be due incredibly small amount of the data. For ABC, it maybe that the weak learner with a max depth of 1 is too weak to predict reliably. In fact, just increasing max depth (unsurprisingly) allowed the model to perform better. It seems a depth of 4 or 5 are the ways to go here.

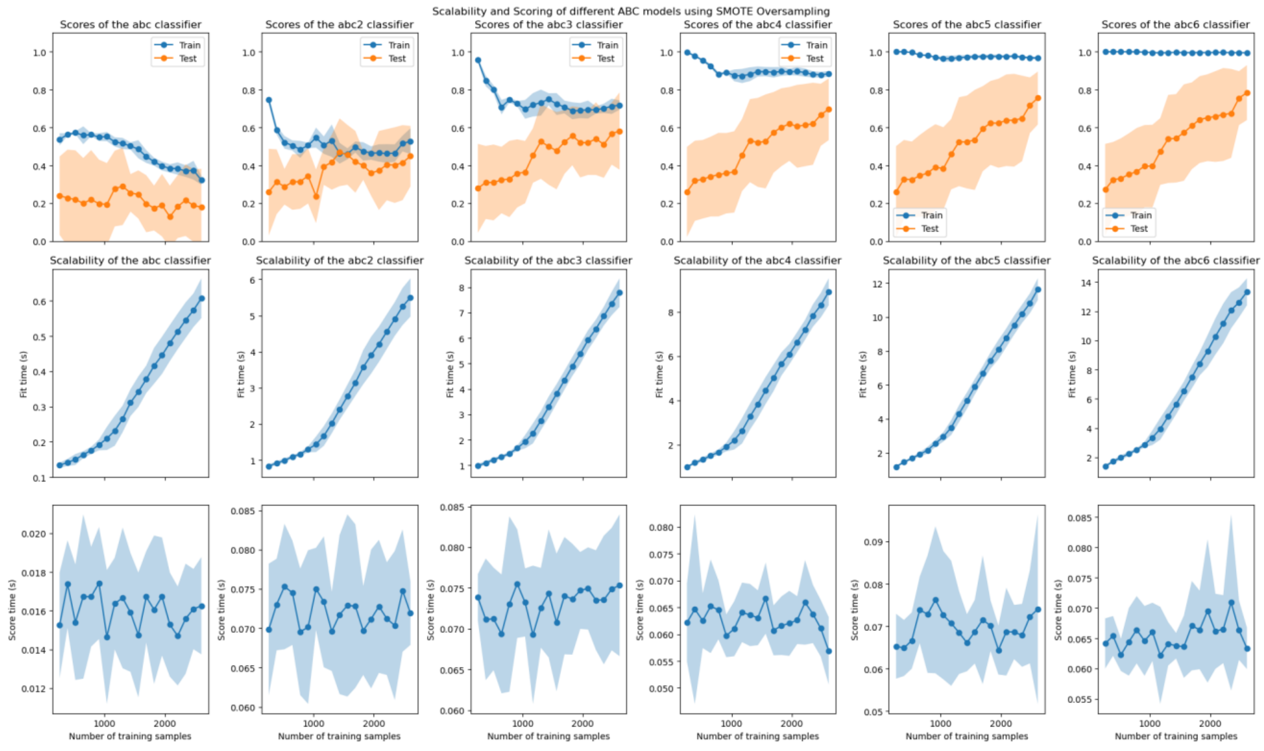


Figure 14: Different max depths for ABC on the Wine Dataset using SMOTE

Mahmoud Shihab
mshihab6@gatech.edu

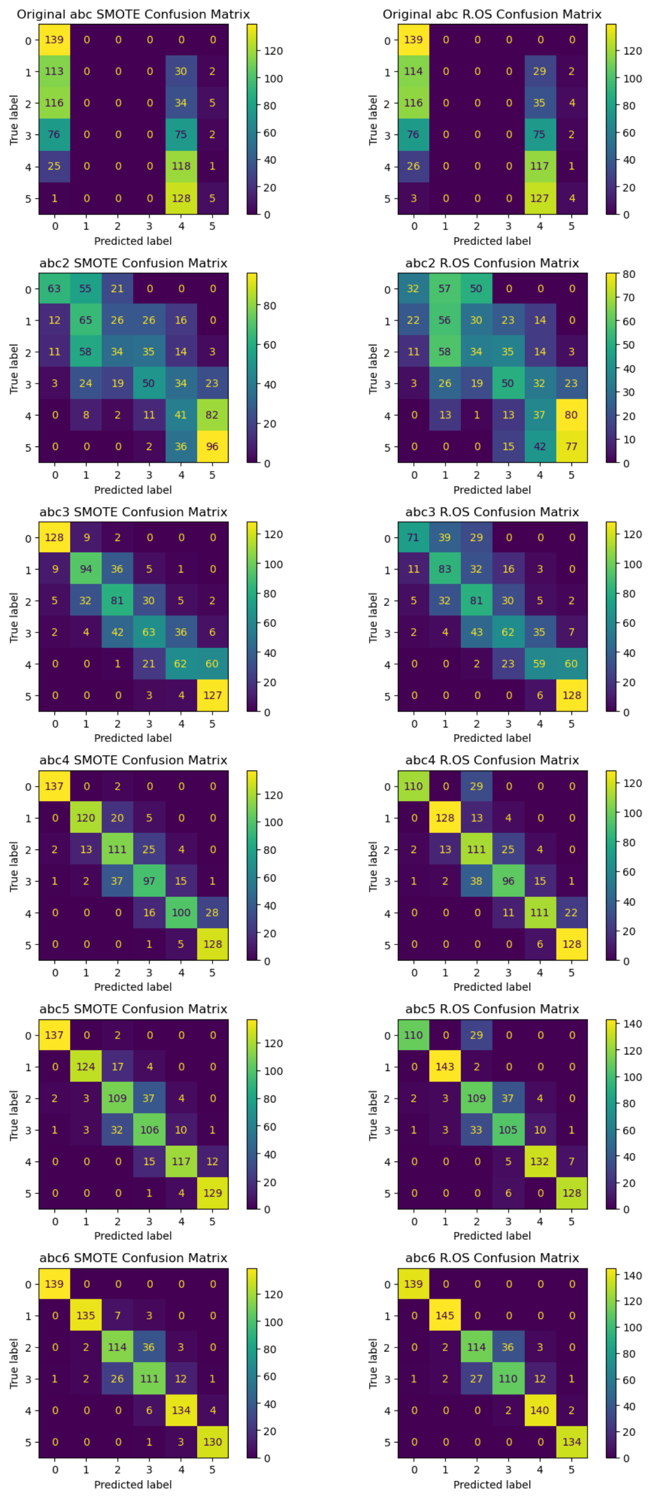


Figure 15: Confusion Matrix for Wine Dataset using SMOTE (Left) and Random Oversampling (Right) for different ABC max depths

For SVC, my suspicions are that either the data requires a bit more cleaning, or the data isn't sufficiently large to support the model. This could also be true for a lot of the models used in this analysis. Unfortunately removing outlies and noise means removing precious data.