Ladan Shamaei (lshamaei3)

ML7641 – Assignment 1 Supervised Learning

Dataset 1) Wisconsin breast cancer dataset from kaggle. This is a classification problem with each instance containing the tumor information and whether it is malignant or benign. Data pre-processing performed includes using label encoder and StandardScaler [2]. The data is imbalanced and since false negatives need to be avoided, therefore recall is a good metric to use.

Dataset 2) Wine quality data from kaggle. This dataset provides wine information with their quality rating between 0-10. Data pre-processing performed includes filling null values with mean, using label encoder, employing MinMaxScaler. 'total sulfur' column was dropped since 'free sulfur dioxide' column conveys the same info. The rated qualities higher than 5 are classified as 1, otherwise they are classified as 0 [1]. The data is imbalanced. F1 is used as metric as F1 is not as sensitive to data imbalance.

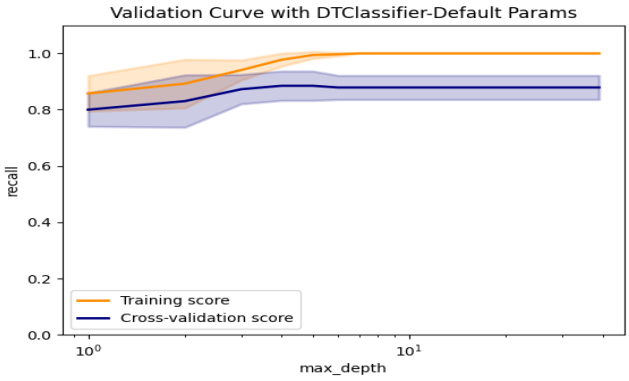
These two datasets are interesting since the learners for the first one all perform differently. The weaker learners perform poorer whereas medium complexity learners perform very good and very complex learners overfit.

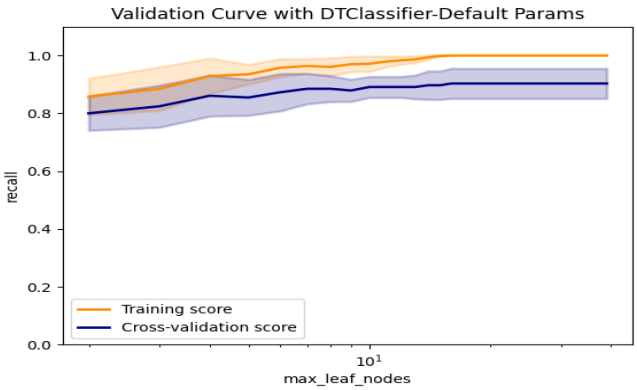
On the other hand, the learners for the second dataset all seem to perform the same. This indicates that the data is simple enough to be learned by weak learners and using more complex learners does not lead to any improvement.

Dataset 1

Decision Tree: Tuned parameters and their final values:

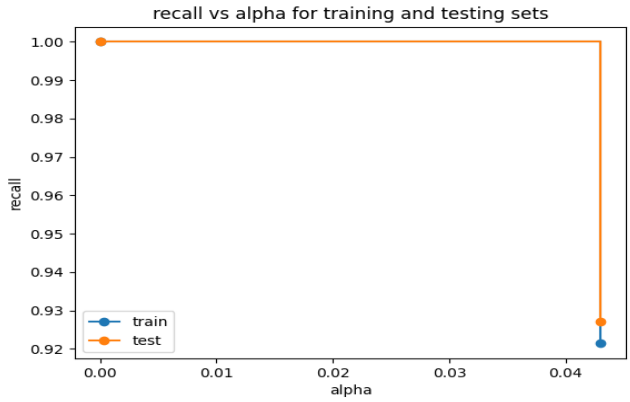
- Criterion = entropy: 'entropy' criterion was used for split quality as shown to perform better using grid search,. This criterion provides a split that results in low entropy.
- leaf size = 3: the validation curve shows that train and validation scores keep improving as leaf sie increases and then they start diverging at max_leaf size of 3, which indicates start of overfitting.
- max depth =3: the validation curve shows that performance keeps getting better by increasing depth up to depth 3. The validation performance degrades slightly after while training score keeps improving, which is an indicator of overfitting.

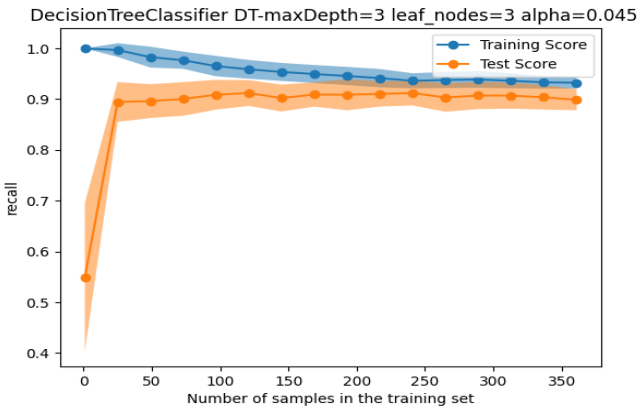




- ccp_alpha = 0.045: Cost complexity parameter curve was generated for both training and validation datasets. The two datasets converge at around alpha = 0.045.

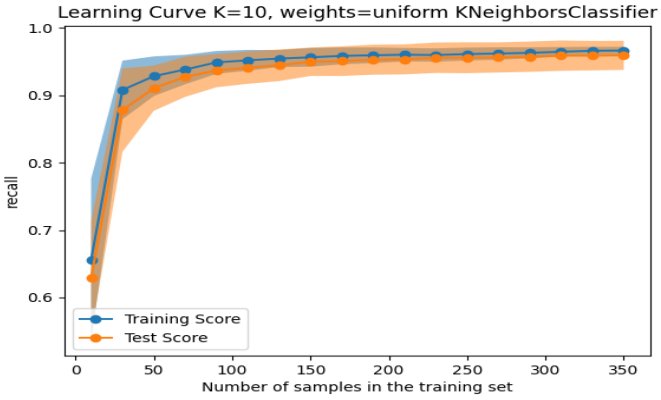
Learning curve for train and testing converge with minimal gap near recall = 0.9. This indicates a good fit with high score and no overfit or underfit issue [4]

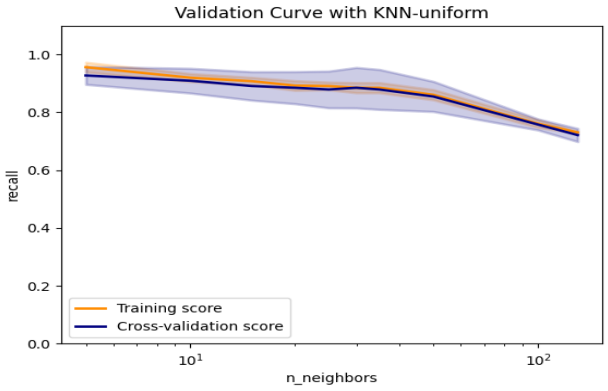


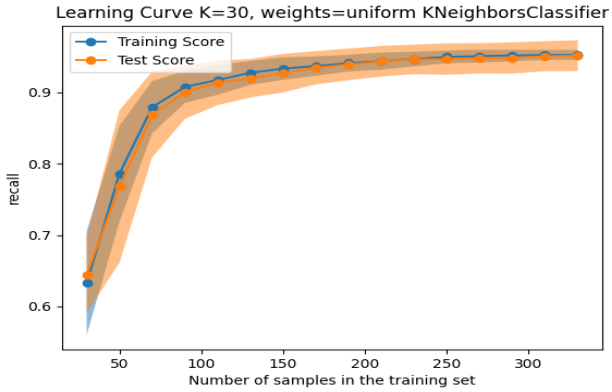


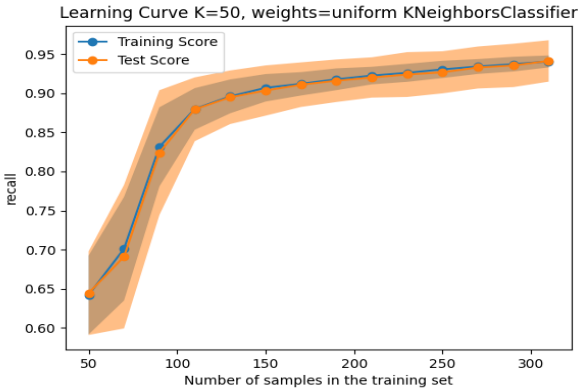
KNN: Parameters used for tuning and their tuned values:

- weights = uniform: 'distance' weights was found to overfit, while "uniform" weights shows a nice convergence of training and testing learning curves.
- When uniform weights is used, validation curve for k<10 shows a small gap between train and test, which indicates overfitting for k<10. The two curves meet at k=10. This value is used for the model. The train and validation curves overlap at higher k's while the model performance goes down for both, which indicates underfitting.







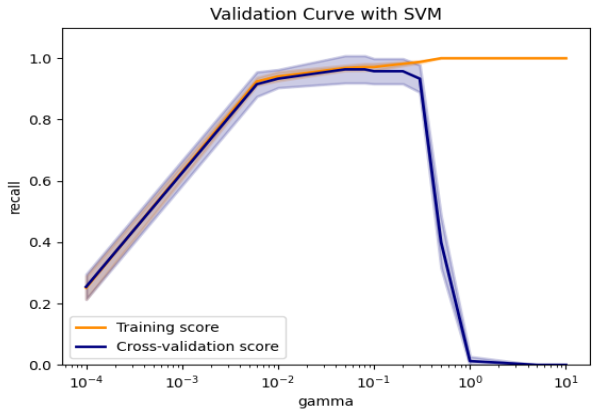


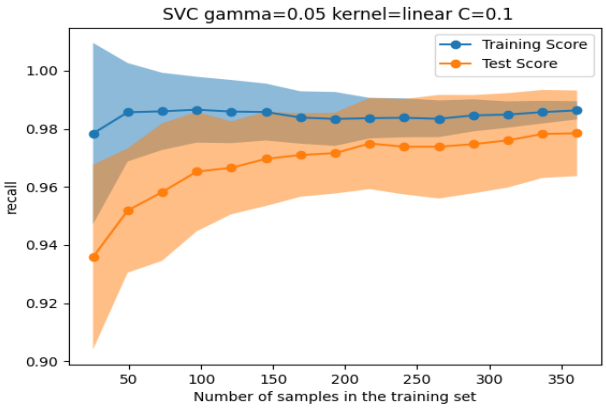
The learning curve above shows that both train and test datasets converge to flat lines as the number of samples increases at a high recall value of 0.95. There is no gap between the train and test curves. This is an indicator of a

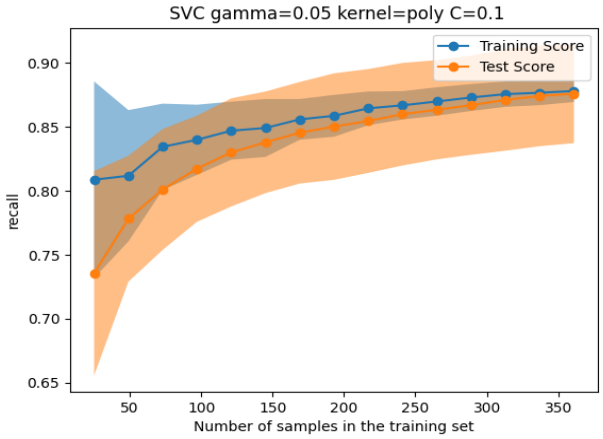
good fit with no overfit or underfit issues. As k increases, the curves show the same trend but the recall decreases for both train and test, which indicates that the model moves towards underfitting.

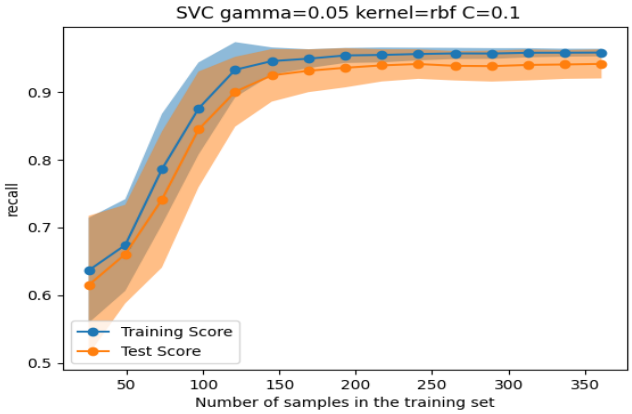
SVM: The parameters tuned with their final values are shown below:

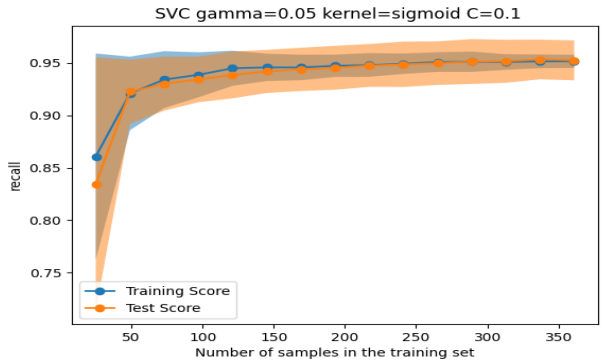
- gamma = 0.05: as the validation curve shows, the performance keeps improving up to gamma = 0.05. After that the validation curve drops significantly while the training curve keeps going up. This indicates overfitting. So gamma value of 0.05 was selected.











Validation Table

kernel	test_recall	train_recall	fit_time	score_time
linear	0.945455	0.968182	0.001017	0.000648
poly	0.648485	0.671212	0.001379	0.000753
rbf	0.878788	0.909091	0.001921	0.001019
sigmoid	0.884848	0.878788	0.002002	0.000895

- Kernel = ‘linear’: Among four kernels evaluated ['linear', 'poly', 'rbf','sigmoid'], linear performs better per the validation table above. It also has lower train and score time. This indicates that the dataset has linear behaviour. Linear kernel is used for the final tuned model.

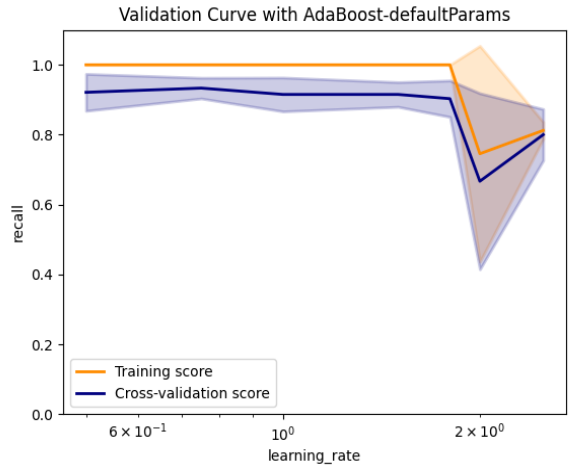
On the learning curve for linear kernel, there is a small gap between the training and test curves. This indicates slight overfit. The test curve is approaching the train curve as the sample size increases, but the test curve doesn't plateau and still has potential to improve and converge with the training curve. Adding more sample data or removing features to make the data simpler may help reducing the gap between the two curves [4].

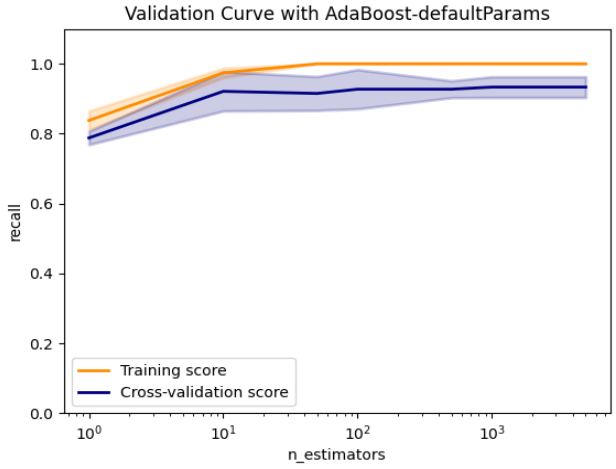
- C = 0.1: Several regularization parameter values were tested. C=0.1 seems to better reduce the variance of recall between the test and train datasets:

C	test_recall	train_recall	fit_time	score_time
0.10	0.951515	0.966667	0.001155	0.000742
0.25	0.951515	0.971212	0.000989	0.000622
0.50	0.951515	0.974242	0.001077	0.000624
0.75	0.933333	0.971212	0.001076	0.000623
1.00	0.945455	0.974242	0.001136	0.000631

Boosting: The parameters tuned and their final values are listed below:

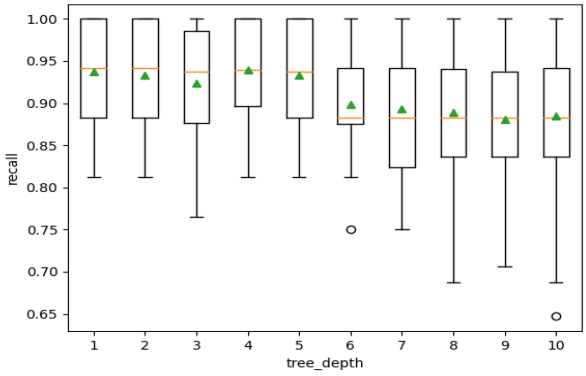
- n_estimators = 100: The validation curve shows improvements up to n_estimator=100 and after that the improvements are not significant.
- learning_rate = 1.8: the model performance drops for learning rates over 1.9. learning_rate of 1.8 was selected

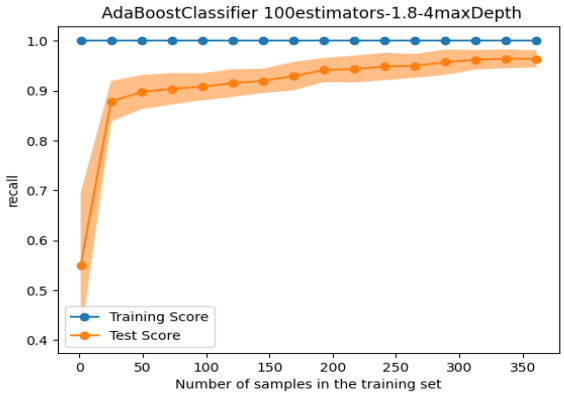




- Max depth = 4: as the graph below shows, max depth of 4 provides better performance.

As the learning curve shows the gap between train and test curves starts getting smaller as the number of samples increases. However, they do not fully converge, which indicates slight overfitting. Adding more training instances will likely reduce the gap between the two curves [4].
The model reaches

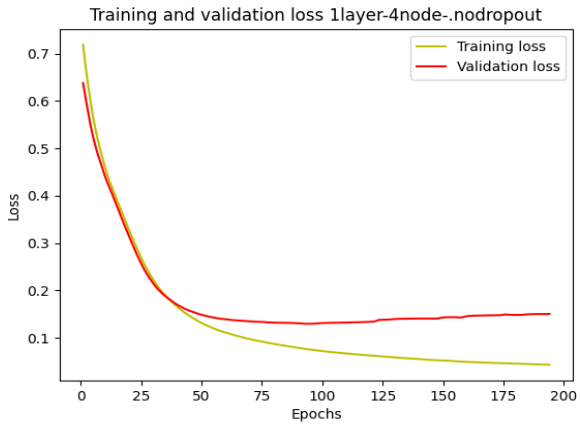


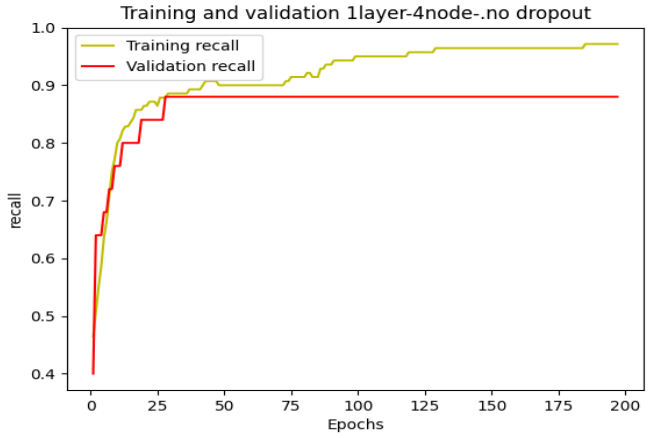


ANN: Tuned parameters and their final values:

- 1 layer – 4 nodes – no dropout – 500 epochs – early stopping – relu for input layer and sigmoid for output layer – binary_crossentropy loss – Adam optimizer – 64 batch size

Even though the simplest network which consists of 1 layer and 4 nodes was used, the loss curve still shows extreme overfitting as there is a gap between the train and validation curves. The model is struggling with scoring the validation data. Changing dropout and learning rate as well as epochs did not help either. This indicates that the dataset is simple and small and neural networks may not a good option for modeling. Neural netwoeks require large datasets to perform well. The max score obtained by the model is 0.9.





Dataset 1 Leaner Comparison and Analysis:

Cross validation: Analyzing the cross-validation results, decision tree and KNN have performed poorly which was expected as they are weak learners. NN has also performed poorly, which is due to the overfitting issue mentioned earlier. However, DT and NN on unseen data have outperformed other learners. This indicates that the test data is significantly different than the train/validation data.

cross validation result

model	val_recall	train_recall	fit_time	score_time
DecisionTreeClassifier	0.787879	0.831818	0.002207	0.000567
SVC	0.957576	0.963636	0.000925	0.000607
AdaBoostClassifier	0.939394	1.000000	0.219791	0.005416
KNeighborsClassifier	0.890909	0.921212	0.000262	0.012327
Sequential	0.885311	0.929876	-	-

unseen data results

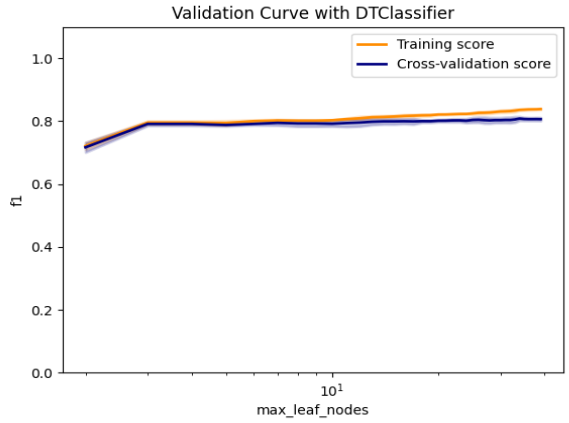
model	test_score	train_time	score_time	dataset
DecisionTreeClassifier	0.978723	0.002937	0.000077	1
SVC	0.957447	0.001105	0.000145	1
AdaBoostClassifier	0.957447	0.359209	0.006800	1
KNeighborsClassifier	0.893617	0.000218	0.103097	1
Sequential	0.957447	38.692021	0.042668	1

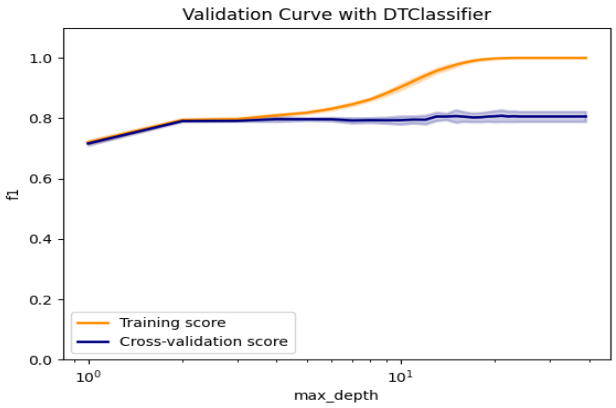
SVC and AdaBoost have shown good performance. AdaBoost takes more run time and memory (higher space and time complexity) compared to SVC. Occam's razor also suggests simpler algorithms over more complex ones when performance is the same. Therefore, SVM is the recommended algorithm. 0.95 is the highest score possible using these algorithms. The 5% error we see is due to the problem/dataset selected here.

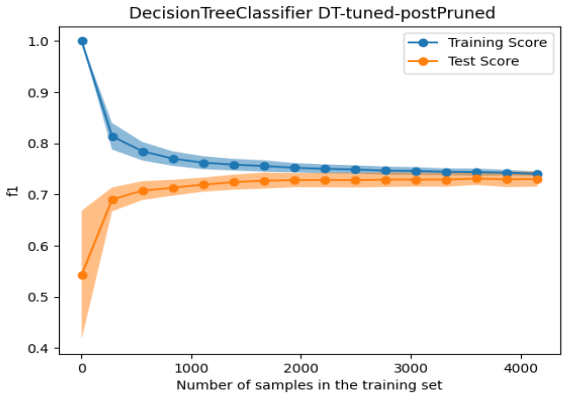
Dataset 2

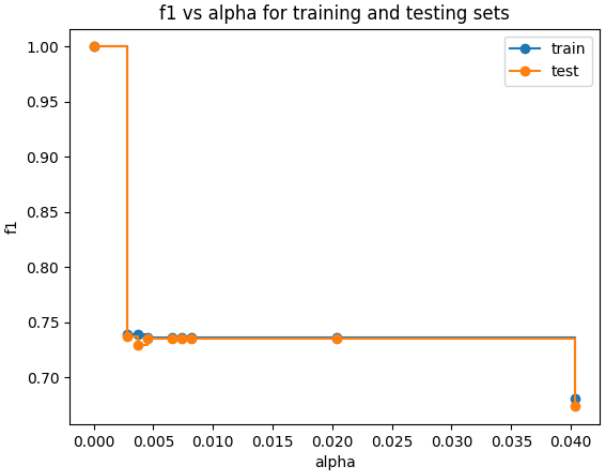
Decision Tree:

- Criterion = entropy: suggested by grid search; separates the split to yield lower entropy.
- max_leaf_nodes = 10: The validation curve shows performance improvement up to max_leaf_nodes = 10 and the train and validation curves start to diverge which indicates overfitting.
- max_depth = 4: The validation curve shows performance improvement up to max_depth = 4 and the train and validation curves start to diverge which indicates overfitting.









Both train and test learning curves converge around 0.75 score, so no overfitting is observed. However, the score is not a high. This indicates that the model has high bias & low variance, so is underfitting. Decision tree is a weak learner and seeing lower performance is expected from this learner. A more complicated model may be able to perform better (more complex learner is explored later in the report).

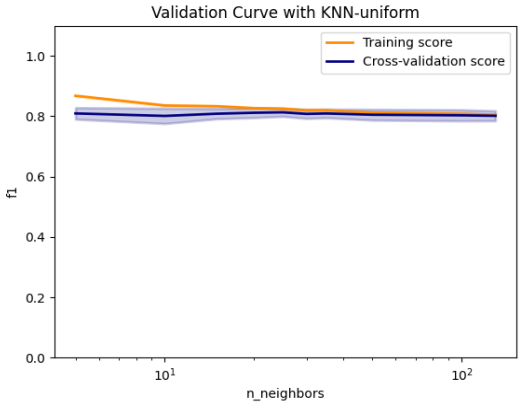
- Ccpa_alpha =0.0025: Post pruning: Plotted validation curve for ccp_alpha. The train and test curve converge at f1 score of 0.75, which lines up with the observations above. The ccp_alpha associated with that score is 0.0025, which was used for the final model.

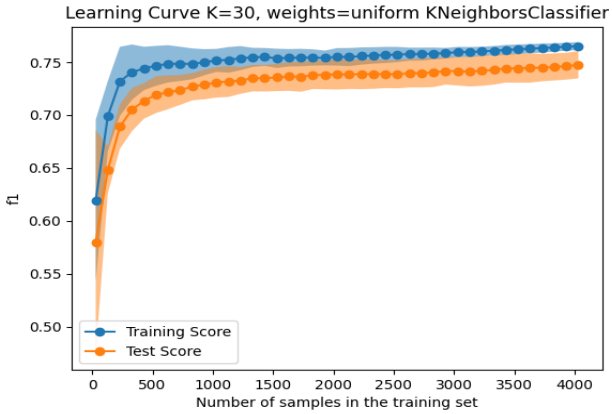
KNN:

- Weights = uniform: When weights='distance' is used, the model shows overfitting. This can be seen in the validation table below too. So weights=uniform was selected.

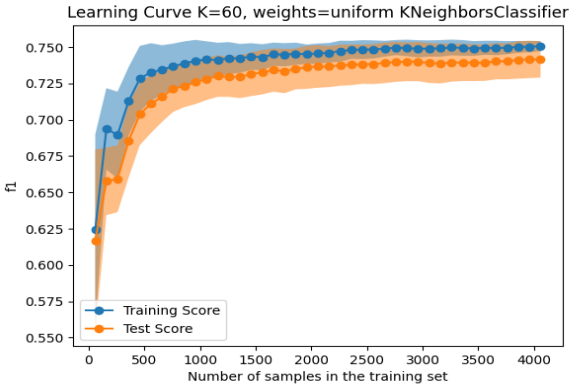
k	weights	test_f1	train_f1	fit_time	score_time
30	uniform	0.806236	0.81835	0.002902	0.038076
30	distance	0.847803	1.00000	0.003031	0.030350

When the weights is changed to uniform, there is a gap between train and validation curves for k smaller than 30. This is the area where overfitting is happening (train dataset performs better than test dataset). The validation curve shows convergence near k=30.





Looking at the learning curve for k=30, there is a small gap between the two, which indicates there is slight overfitting issue. Two larger k values were explored to reduce the gap between the two lines. The gap is reduced slightly when k is doubled (k=60). However, the change is not significant. Plus, the f1 score goes down for both train and validation (move towards underfitting). So k value was maintained at 30. To reduce the gap, one solution is to add more train instances as the test curve has the potential to get closer to train curve as the sample size increases [4].



Similar to DT learner, when looking at learning curve, **both** train and test curves score around 0.75, which is not a very high score. Since the train and test curve converge at a lower score, the model has high bias and low variance issue, underfitting the training data. One solution to improve the model is to increase the features, which is not feasible here. The other solution is to use more complex algorithms (more complex learner is explored later in the report).. KNN, which is a lazy learner, can perform lower than anticipated.

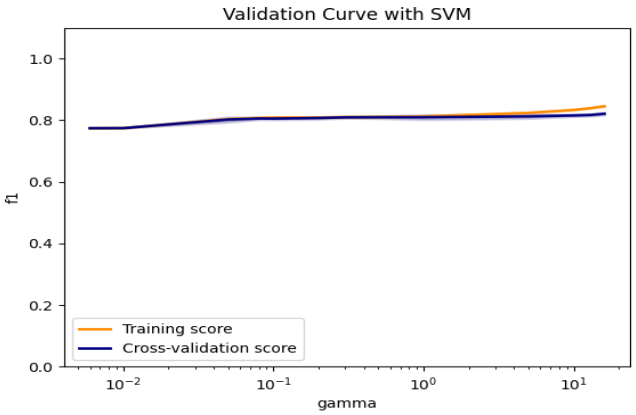
- Metric = euclidean: Validation table for four metrics ['manhattan', 'cosine', 'minkowski', 'euclidean'] indicates better performance and lower fit/score time for euclidean distance metric.

k	distance_metric	test_f1	train_f1	fit_time	score_time
30	manhattan	0.805916	0.820880	0.003151	0.055937
30	cosine	0.809563	0.818837	0.000743	0.144174
30	minkowski	0.805004	0.819271	0.003393	0.048913
30	euclidean	0.810725	0.818574	0.003105	0.043339

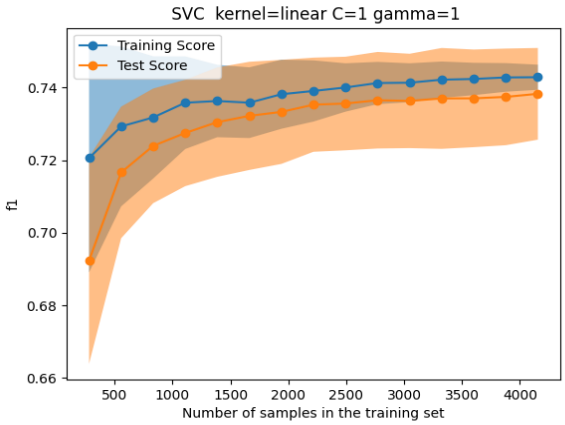
SVM: Tuned parameters and their final values:

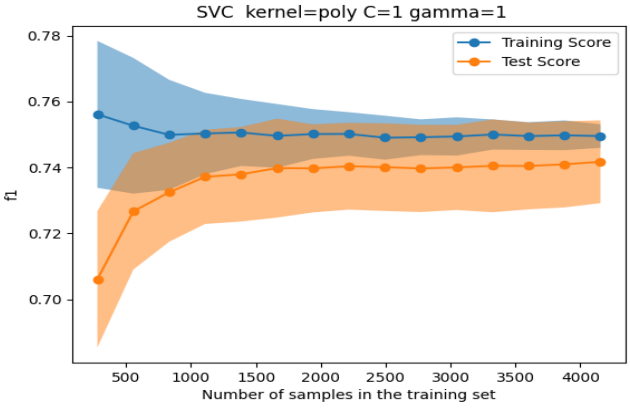
- gamma = 1: Validation curve for gamma (below) shows improved performance up to gamma value of 1, after which a divergence is observed between the training and validation curve, which shows transition to overfitting. Gamma of 1 is used.

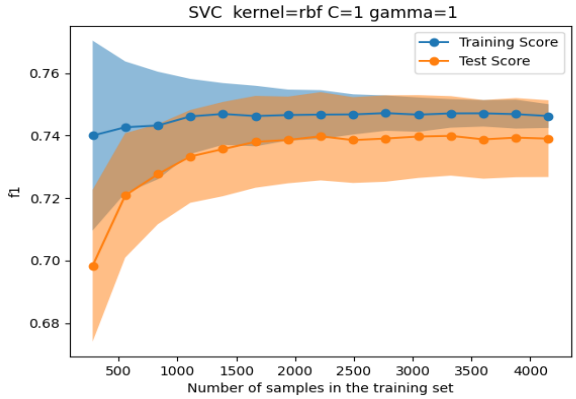
Kernel = linear: Validation table for four kernels ['linear', 'poly', 'rbf','sigmoid'] shows similar performance for linear, poly and rbf. However, sigmoid is performing poorly. This can also be seen in the sigmoid learning curve below, where the algorithm is not able to learn the data. Among other three kernels, linear has shorter train and score time. In addition, the learning curve for linear kernel shows convergence of train and test curves, indicating that there is no overfitting issue where as rbf and poly show a small gap between train and test curves. For these reasons, linear kernel was selected.

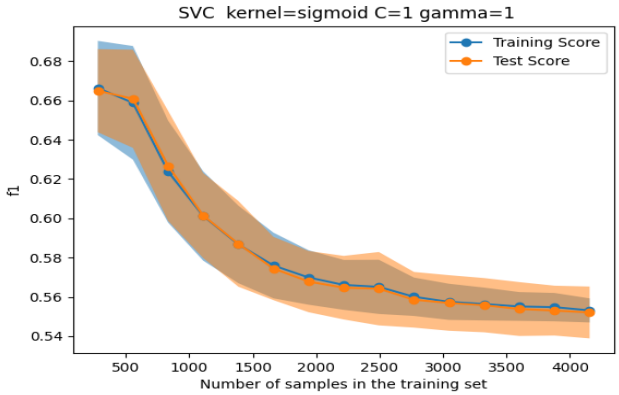


kernel	test_f1	train_f1	fit_time	score_time
linear	0.807513	0.808611	0.176735	0.027162
poly	0.807729	0.811754	0.179362	0.029223
rbf	0.809822	0.812712	0.230086	0.060096
sigmoid	0.649683	0.651432	0.371642	0.060632





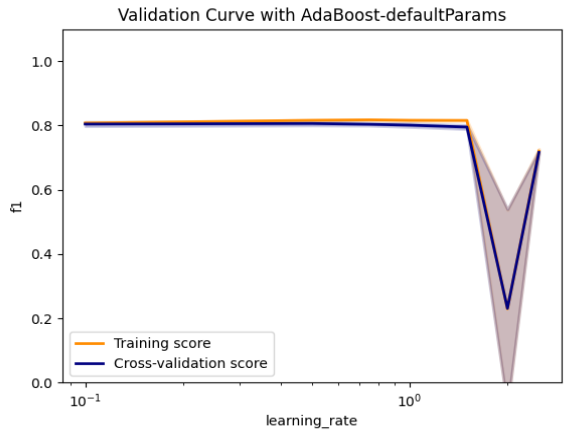


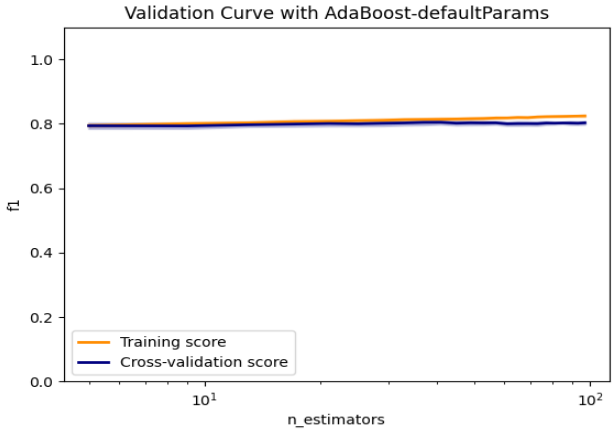


Although the train and test learning curves converge, the score is around 0.74, which means that the learner has not learned the data well. A more complex model may perform better for this data (more complex learner is explored later in the report).

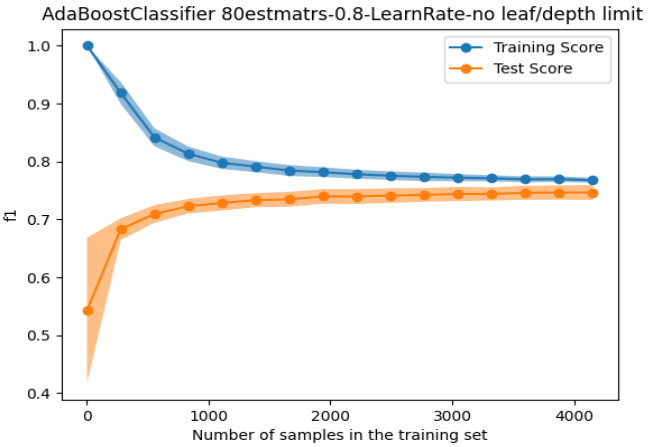
Boosting: The parameters tuned and their final values are listed below:

- Learning rate = 0.8 - the validation curve below shows the divergence between training and validation curves at 0.8 learning rate, which indicates overfitting at learning rates above 0.8. This value is used for the model.
- n_estimators = 80 – the validation curve below shows that the train and validation curves diverge at around n_estimators=80. In other words, overfitting to be expected if n_estimators>80.



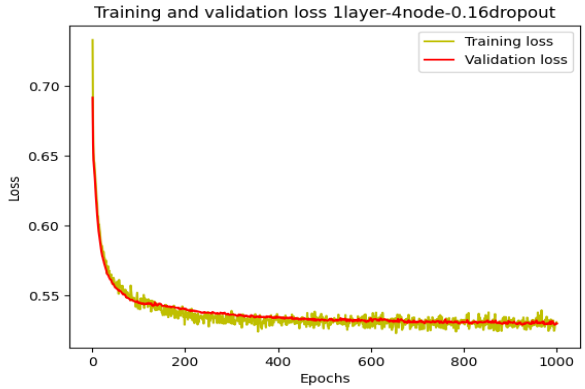


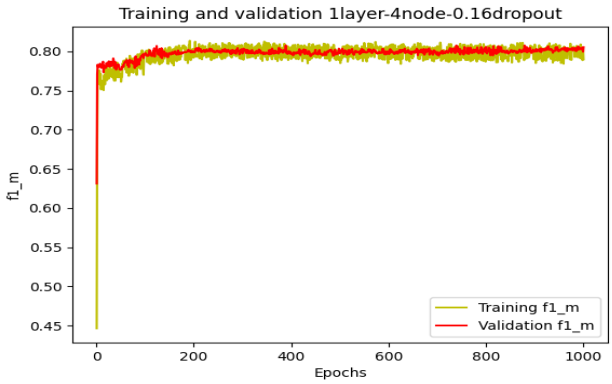
- Max_depth and leaf_nodes: Analyzing the learning curve below, there is no overfitting issue as the train and test curves converge. So no max_depth or leaf_nodes need to be specified. Unlike decision tree where overfitting was observed when no max_depth or max_leaf_nodes were specified, boosted tree model does not show overfitting even with no max_leaf and max_depth specified. This was expected as 80 estimators are used which minimizes overfitting.



The learning curve indicates training and test convergence at around f1 = 0.75. This is in line with the score observed for DT and KNN and SVM. However, AdaBoost is a more complex learner and is expected to outperform simple models. This could be an indicator of error of the dataset and irreducible error.

ANN: Tuned parameters and their final values
1 layer – 4 nodes – 0.16 dropout – Relu activation for input layer and sigmoid for output layer – binary_crossentropy loss – 1000 epochs – 64 batch size -Adam optimizer - early stopping
The loss curve of validation and training are close to each other which indicates there is no overfit or underfit issues. The f1 score shows slight improvement compared to the other four leaners, which was expected as this is a more





Dataset 2 Leaner Comparison and Analysis:

Cross validation results indicate learners performing in the same ball park. DT performs slightly lower than the other four, which is because it is a weak learner. Considering that the other four learners perform in the same ball park, more simple learners are suggested to be selected to save on run time and memory and also follow the Occam razor's suggestion. 0.8 is the highest we can score with these models. This indicates that 20% of the error is associated with the data/problem selected here. Adaboost performs slightly better for unseen data. However, the run time and memory usage will be a lot higher. KNN seems to be a good model to use.

cross validation result

model	val_f1	train_f1	fit_time	score_time
DecisionTreeClassifier	0.787646	0.800898	0.005432	0.000888
SVC	0.807225	0.808200	0.178116	0.027111
AdaBoostClassifier	0.803992	1.000000	0.017052	0.001019
KNeighborsClassifier	0.809357	0.818477	0.003038	0.044596
Sequential	0.803866	0.799295	-	-

unseen data results

model	test_score	train_time	score_time	dataset
DecisionTreeClassifier	0.803204	0.006318	0.000116	2
SVC	0.815623	0.307749	0.039565	2
AdaBoostClassifier	0.834043	0.021400	0.000311	2
KNeighborsClassifier	0.812537	0.003172	0.049557	2
Sequential	0.813020	59.514676	0.068161	2

Comparing Two Datasets:

Dataset 1 is a smaller and simpler data to learn. Most learners were able to learn the data by lower % error. By using slightly complex algorithm such as SVM, high performance was achieved. However, second dataset had higher % error. The performance was lower compared to the first dataset for all the learners. To help with the performance improvement, one solution is adding more features to the dataset if possible. The other solution is to decrease the regularization. This solution was explored by changing the learning rate, adding layers/neurons, reducing k in KNN, adding more learners to AdaBoost, less pruning etc. However, the learners' performance did not seem to improve, which indicates that the error is due to the data/problem and is not reducible [4].

References

1. https://www.geeksforgeeks.org/wine-quality-prediction-machine-learning/
2. https://towardsdatascience.com/building-a-simple-machine-learning-model-on-breast-cancer-data-eca4b3b99fa3
3. scikit-learn.org
4. https://www.dataquest.io/blog/learning-curves-machine-learning/
5. https://medium.com/@chaudhurysrijani/tuning-of-adaboost-with-computational-complexity-8727d01a9d20
6. https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model