CS7641 Assignment 1 - Eric Gregori

Classification Problems

WiFi Localization [3]

"Collected in indoor space by observing signal strengths of seven WiFi signals visible on a smartphone. The decision variable is one of the four rooms. Each attribute is wifi signal strength observed on smartphone." [9]

File: wifi_localization.txt Attributes: 7 Categories: 1,2,3,4 Instances: 2000

Attributes are balanced: min=-98.0, max=-10.0

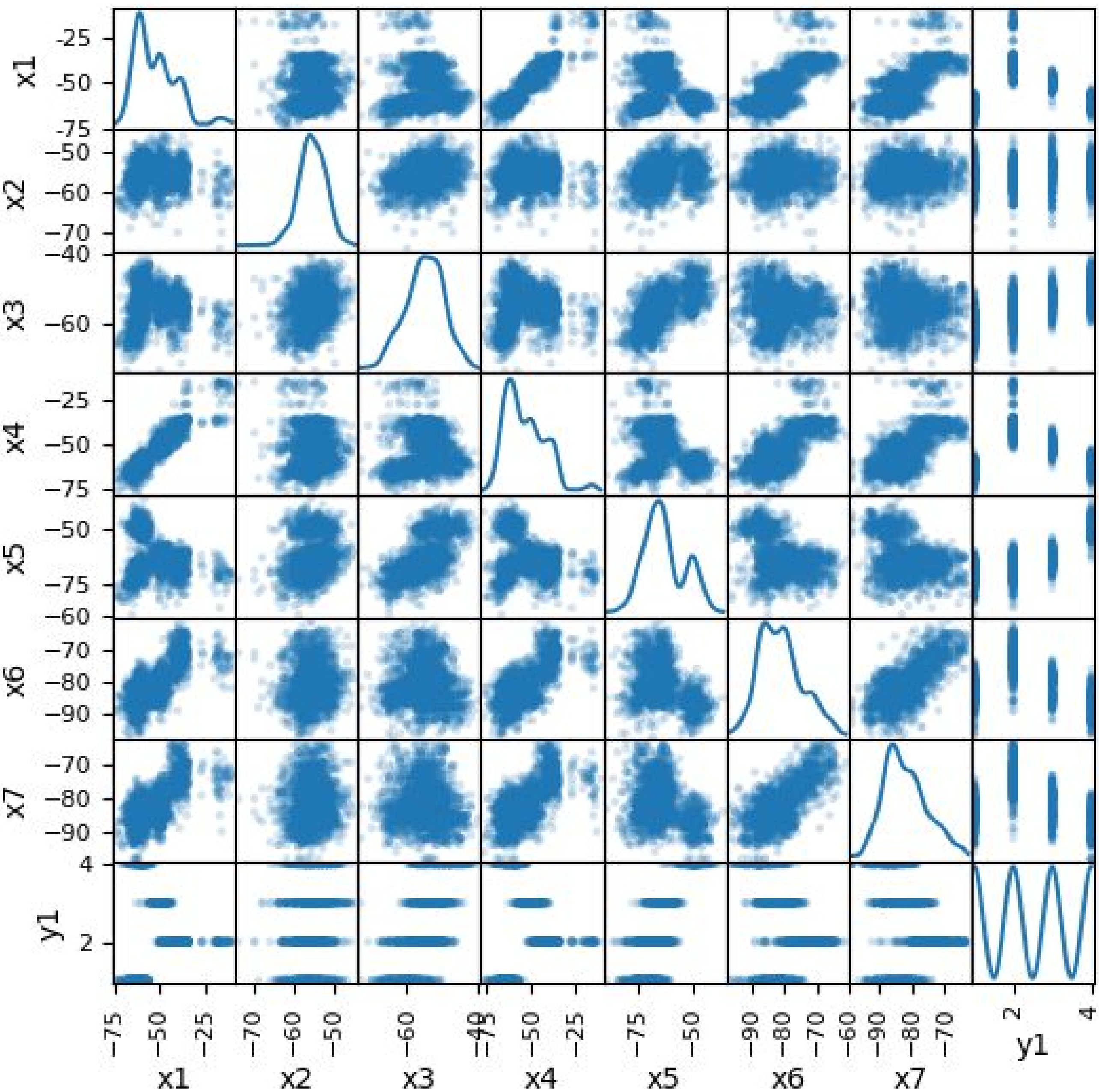


Figure 1: Scatter matrix with KDE diagonal of wifi_localization.txt

Letter Recognition [2]

"Database of character image features; try to identify the letter" [8]

File: letter-recognition.data   Attributes: 16   Categories: 26 (letter)   Instances: 20000

Attributes are balanced: min=0, max=15

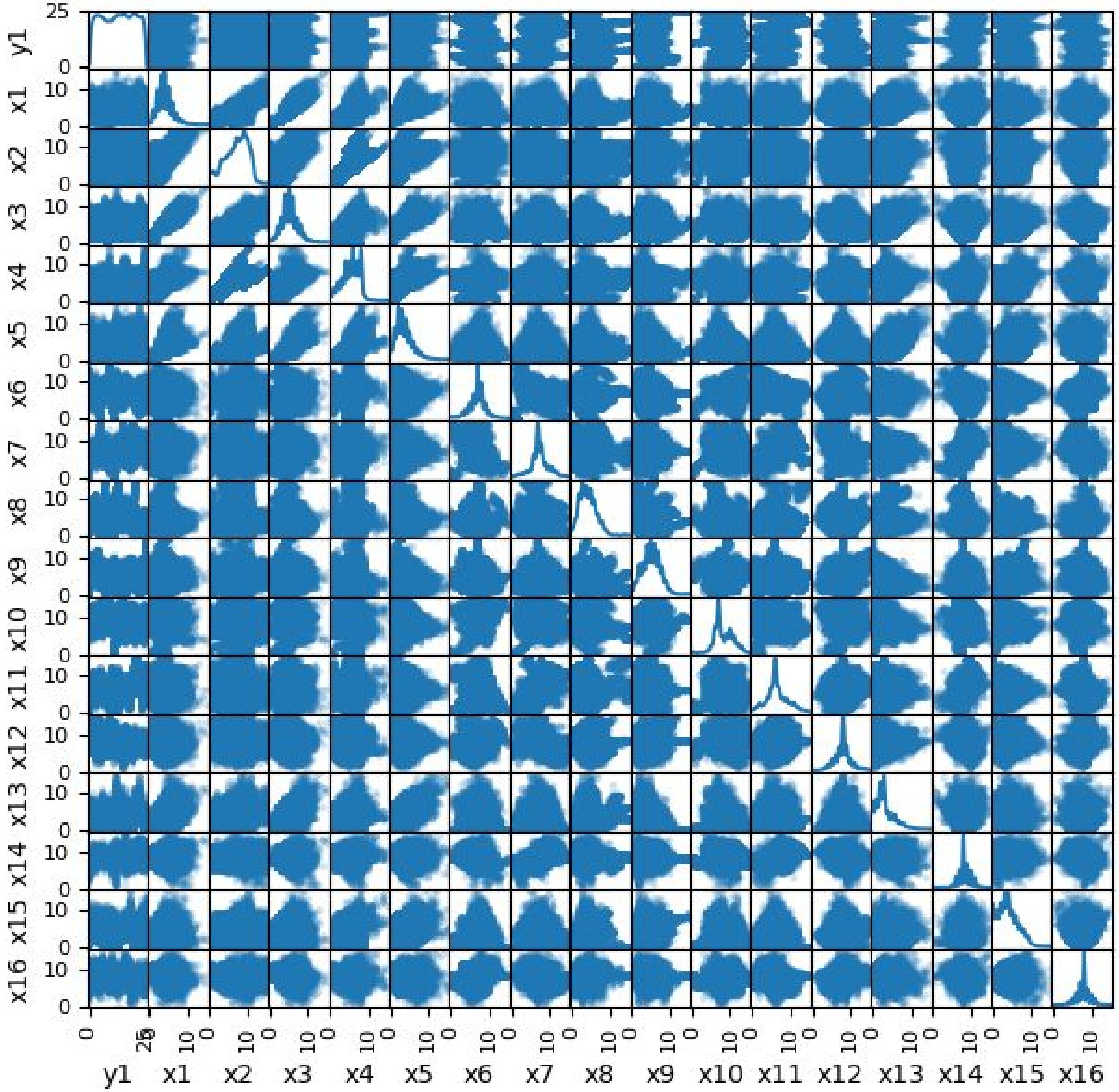


Figure 2: Scatter matrix with KDE diagonal of letter-recognition.data

Kernel Density Estimation [9] is used to plot the distribution for each attribute down the diagonal of the scatter plot. Note: X2 and X3 in the WiFi dataset shows a Gaussian like distribution indicating the attribute data may be random and of little value. The wifi attribute range if from -98 to -10. The letter attribute range is between 0 and 15. Both datasets contain balanced attributes (the attributes are all within the same range). Some classifiers will require the attributes to be scaled from -1.0 to 1.0.

Why Are the Datasets Interesting?

This analysis is based only on the scatter matrix. A second 'interesting' analysis at the end of this paper will include the results from the classifications. Based on the scatter matrix's, the wifi dataset (figure 1) does not look very interesting. Attributes X2 and X3 may be random providing very little information, and X6 and X7 may correlate meaning one of them will not provide any value. The letter dataset (figure 2) looks more interesting. There does not appear to be any attributes containing random data or any strong correlation between attributes (with the exception of possibly X1 and X3).

Testing/Data Collection

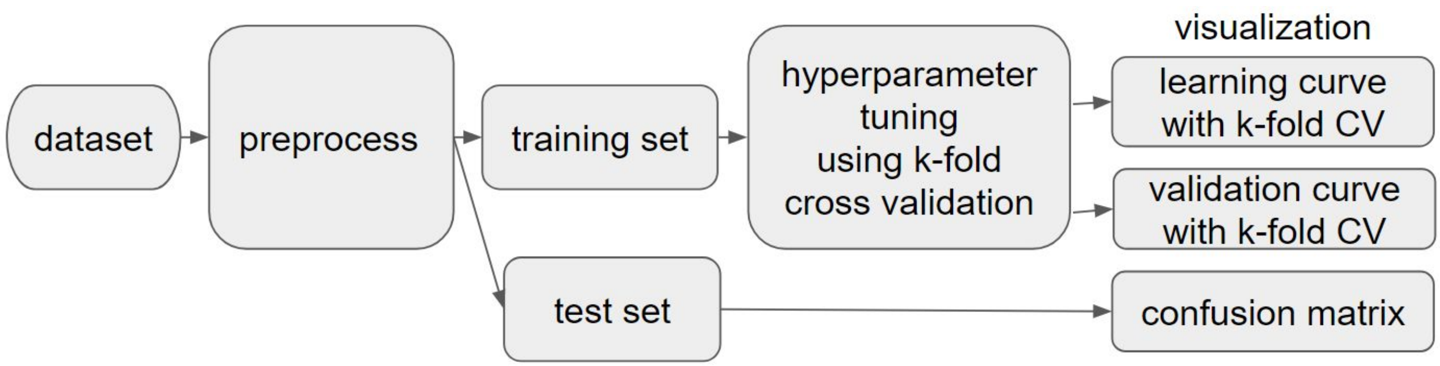


Figure 3. Data collection pipeline applied to each algorithm

The dataset is preprocessed to scale the attributes between -1 and +1. The classes are converted to integers. The resulting dataset is split into a training set and test set. The function sklearn.model_selection.GridSearchCV() [10] is used to find the best hyperparameters using k-fold cross validation. K-fold cv splits the training data into validation and training data. *"... by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets. A solution to this problem is a procedure called cross-validation (CV for short). A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called k-fold CV, the training set is split into k smaller sets (other approaches are described below, but generally follow the same principles)."*[11]

The sklearn.model_selection.learning_curve() function *"determines cross-validated training and test scores for different training set sizes. A cross-validation generator splits the whole dataset k times in training and test data. Subsets of the training set with varying sizes will be used to train the estimator and a score for each training subset size and the test set will be computed. Afterwards, the scores will be averaged over all k runs for each training subset size."* [12]

The sklearn.model_selection.validation_curve() function *"determine training and test scores for varying parameter values. Compute scores for an estimator with different values of a specified parameter. This is similar to grid search with one parameter. However, this will also compute training scores and is merely a utility for plotting the results.*" [13]

Finally, the confusion matrix plots the values predicted from the trained classifier with the test set. Along the diagonal it shows valid or true classifications. Outside the diagonal are classification error.

Decision Trees (DT) [4] - sklearn.tree.DecisionTreeClassifier()

The scikit-learn DecisionTreeClassifier() supports pruning during tree generation by providing parameters for limits to tree depth and limits to various splitting decisions.

Parameter	Description	Tested Range
criterion	The function to measure the quality of a split	'gini','entropy'
splitter	The strategy used to choose the split at each node	'best','random'
max_features	The number of features to consider when looking for the best split	,'auto','sqrt','log2',None
max_depth	The maximum depth of the tree	3,4,5,6,None
min_samples_split	The minimum number of samples required to split an internal node	2,3,4
min_samples_leaf	The minimum number of samples required to be at a leaf node	1,2,3,4

Boosting (ADA) (AdaBoost with DecisionTreeClassifier()) [5]

Parameter	Description	Tested Range
n_estimators	The maximum number of estimators at which boosting is terminated (int)	10 to 100
DTC	See decision tree parameters above	

Neural Network (ANN) (Multi-Layer Perceptron) [6] - sklearn.neural_network.MLPClassifier

Parameter	Description	Tested Range
hidden_layer_sizes	The ith element represents the number of neurons in the ith hidden layer (tuple)	10, 50, 100
activation	Activation function for the hidden layer.	'identity', 'logistic', 'tanh', 'relu'
solver	The solver for weight optimization (string)	'lbfgs', 'sgd', 'adam'
learning_rate	Learning rate schedule for weight updates (string)	'constant', 'invscaling', 'adaptive'
learning_rate_init	It controls the step-size in updating the weights (float)	0.001, 0.010
max_iter	Maximum number of iterations (int)	150, 200, 250

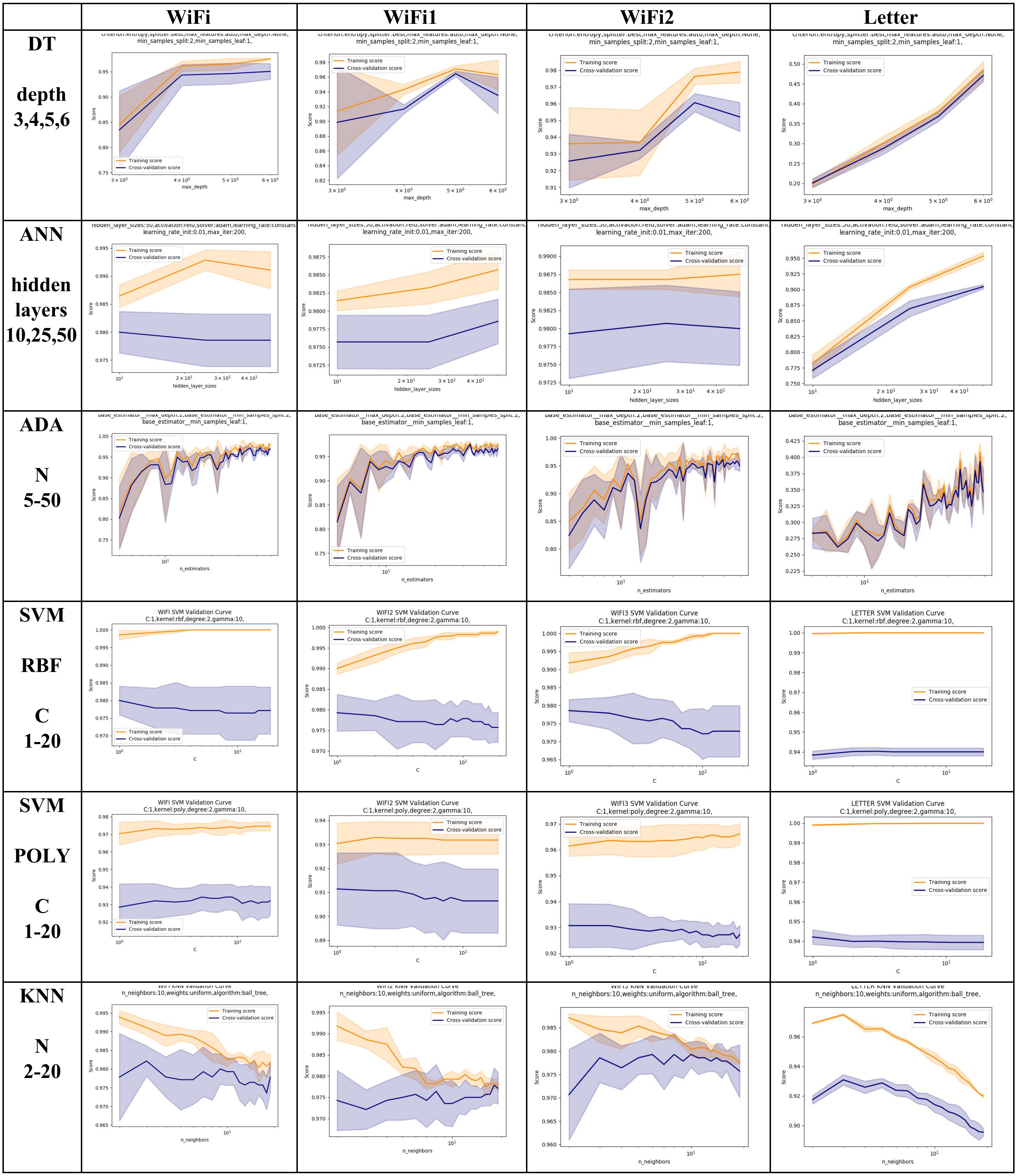
Support Vector Machines (SVM) [7] - sklearn.svm.SVC

Parameter	Description	Tested Range
C	Penalty parameter C of the error term (float)	1 to 10
kernel	Specifies the kernel type to be used in the algorithm	'rbf''poly'
degree	Degree of the polynomial kernel function ('poly') (int)	1,2,3,4,5,6
gamma	Kernel coefficient for 'rbf', 'poly' and 'sigmoid' (float)	0,10,100

k-nearest neighbors (KNN) [8] - sklearn.neighbors.KNeighborsClassifier

Parameter	Description	Tested Range
n_neighbors	Number of neighbors to use by default for k_neighbors queries (int)	2-10
weights	weight function used in prediction	'uniform', 'distance'
algorithm	Algorithm used to compute the nearest neighbors	'ball_tree', 'kd_tree', 'brute'

Manual HyperParameter Tuning - Score = Accuracy

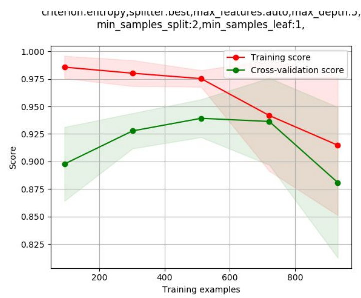
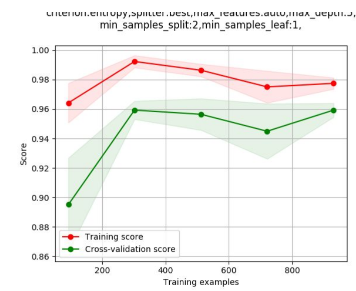
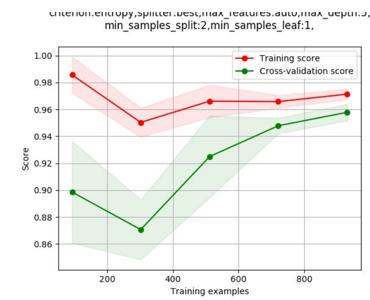
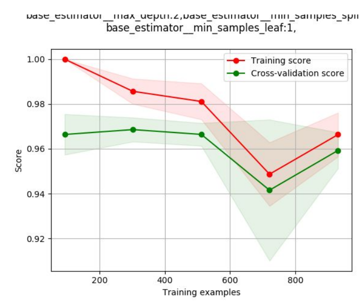
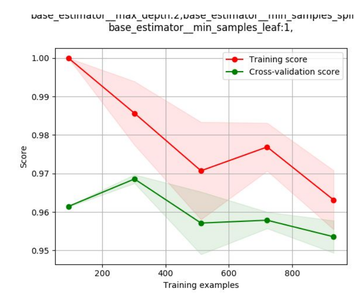
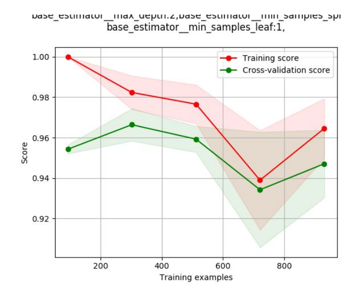
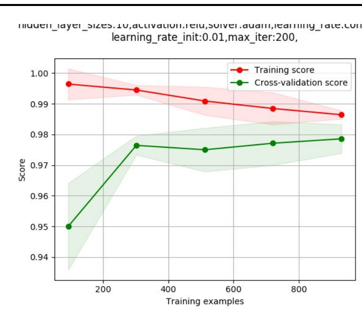
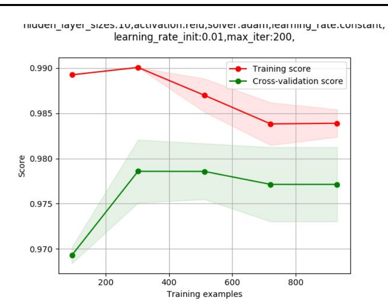
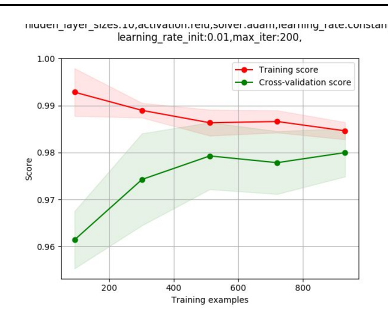
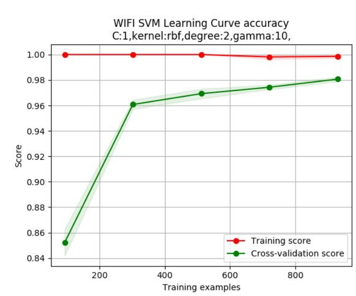
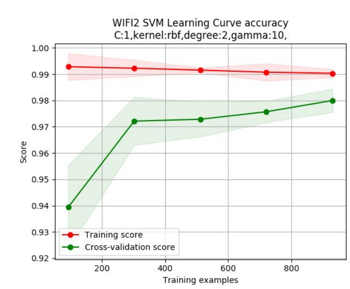
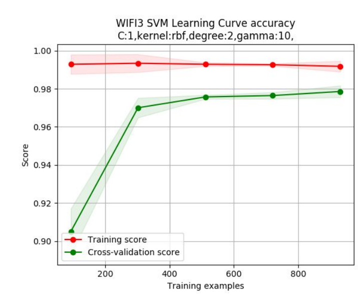
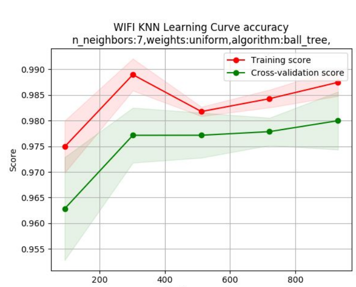
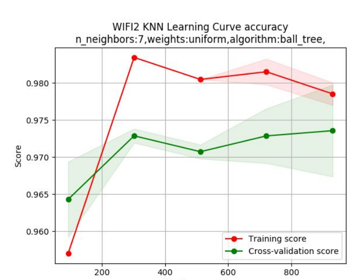
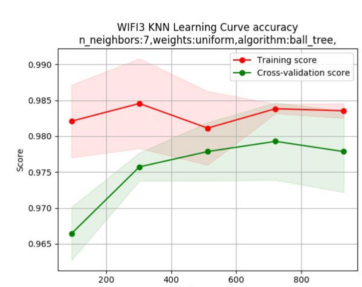


Validation Curves: Plot of Training and Cross-Validation score while changing a single hyperparameter

Training = Orange, Cross-Validation = Purple, Score = Accuracy

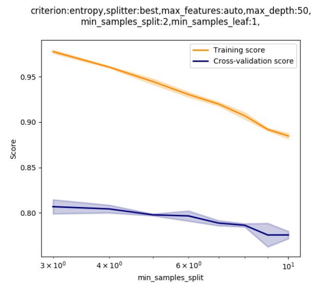
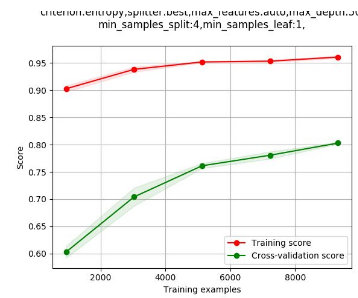
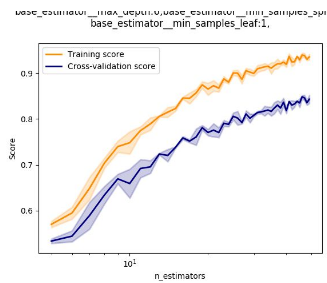
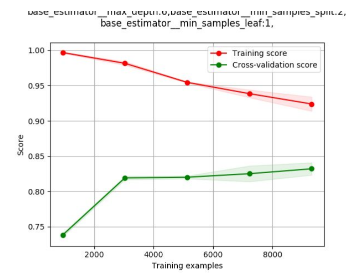
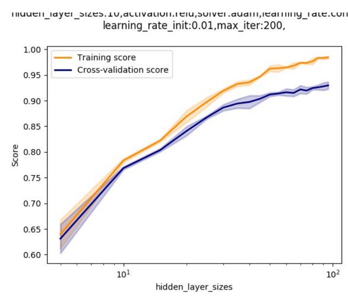
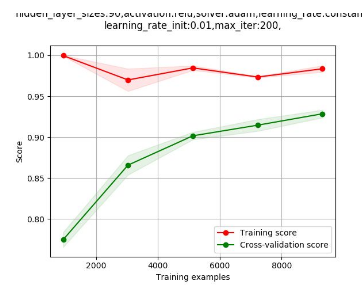
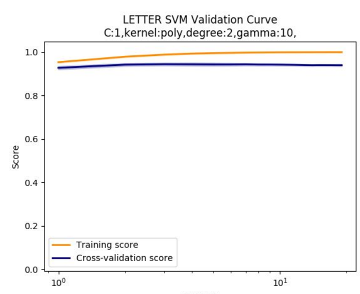
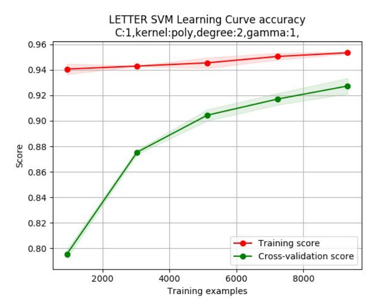
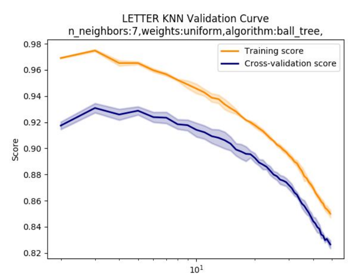
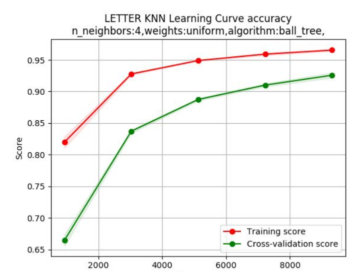
WiFi = X1-X7, WiFi1 = X4,X5,X6,X7 (no X1,X2,X3), WiFi2 = X1,X2,X3,X4,X5,X6 (no X7)

WiFi Results

Alg/Param	WIFI	WIFI2	WIFI3
DT entropy best auto depth:5 ms_split:2 ms_leaf:1			
ADA **N = 35** entropy best sqrt depth:2 ms_split:2 ms_leaf:1			
ANN 10 Relu Adam Constant 0.010 200			
SVM 1 Rbf 2 10			
KNN 7 Uniform ball_tree			

Learning Curves: Training = Red. Cross-Validation = Green, Score = accuracy

Letter Results

<table>
<tr><th>Alg/Param</th><th>Val Range</th><th>Letter Validation</th><th>Letter Learning</th></tr>
<tr><td>DT
entropy
best
auto
depth:50
ms_split:4
ms_leaf:1</td><td>ms_split
3-10</td><td>
</td><td>
</td></tr>
<tr><td>ADA
60
entropy
best
sqrt
depth:6
ms_split:2
ms_leaf:1</td><td>N
5-50</td><td>
</td><td>
</td></tr>
<tr><td>ANN
90
relu
adam
constant
0.010
200</td><td>hidden
layer
Size
5 - 100</td><td>
</td><td>
</td></tr>
<tr><td>SVM
1
poly
2
1</td><td>gamma
0-20</td><td>
</td><td>
</td></tr>
<tr><td>KNN
7
Uniform
ball_tree</td><td>N
2-50</td><td>
</td><td>
</td></tr>
</table>

Learning Curves: Training = Red. Cross-Validation = Green, Score = accuracy

Validation Curves: Training = Orange, Cross-Validation = Purple, Score = Accuracy

Analysis

Algorithm Comparison

	DT	ANN	ADA	SVM	KNN
# hyper parms	6	2*	1	4,5	3
scaling	Not Required	[-1,1]	Not Required	[-1,1]	Not Required*
training time	10’s of ms	seconds	seconds	seconds	10’s ms
prediction time	ms	ms	100’s of ms	seconds	100’s ms

The ideal classifier requires no tuning and can learn anything quickly. The more hyperparameters, the harder and longer it can take to tune a model. AdaBoost was the simplest to tune with only one parameter. Although the underlying decision tree has many parameters, very little tuning of the underlying decision tree is required because it only has to classify better than chance. The hardest to tune was the stand alone decision tree classifier (DT). With 6 hyperparameters each one with a dependency on the others, tuning the decision tree required many validation plots to get correct.

Although the ANN has 6 hyperparameters, most are categorical as opposed to numeric. This decreases the the total number of possible combination that need to be tested. The ANN also appears to be the most forgiving of poor hyperparameter choices when it comes to classification. Poor hyperparameter choices are paid for in learning time.

An ideal classifier can learn anything. The DT, ADA, and KNN classifiers were the most tolerant of unscaled data. Both datasets for this report were balanced (all the attributes in the same range). The KNN requires balanced data.

Algorithm training and prediction time depend heavily on the dataset and hyperparameters. Instead of listing exact times relative times are provided. These times were measured while training and fitting to the datasets in this report. Generally, classifiers are trained once and used many. For most applications the ability to predict quickly is more important than training times. The DT and ANN are very quick to provide predictions. This makes sense because the number of computations when fitting for these algorithms is relatively low. The SVM is slow in training and predicting. Most likely due to the number of complex computations required to calculate a prediction.

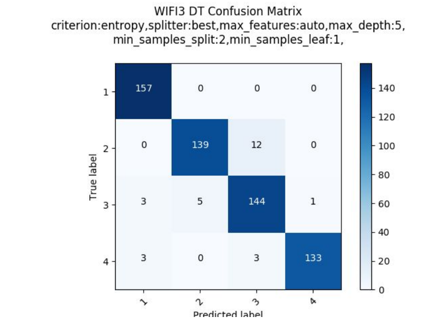
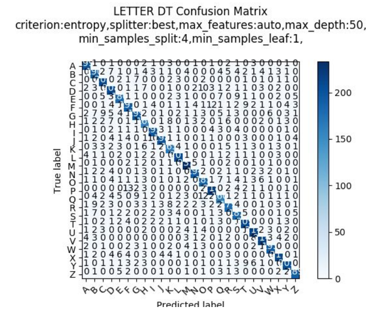
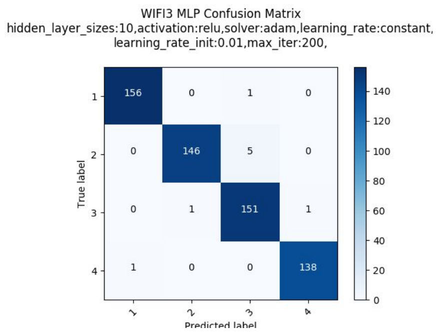
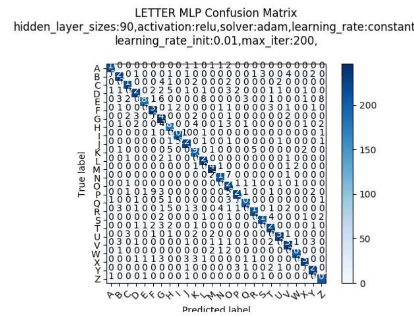
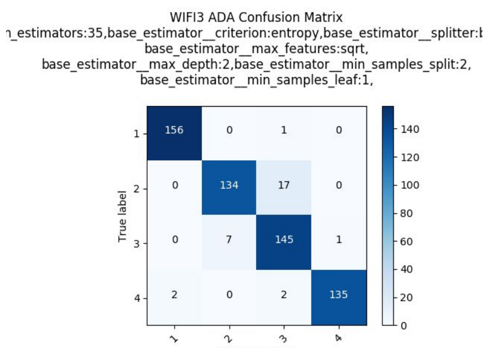
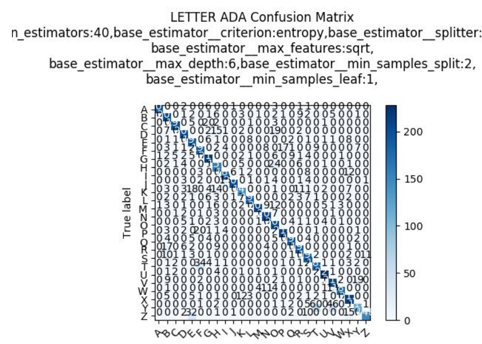
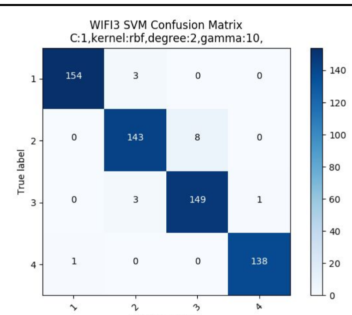
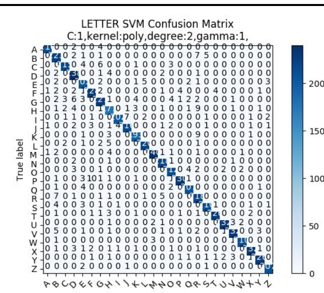
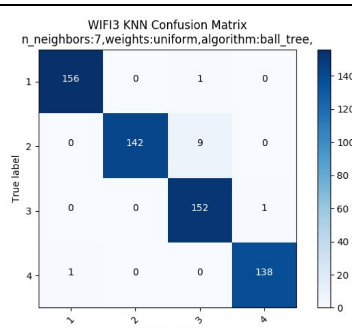
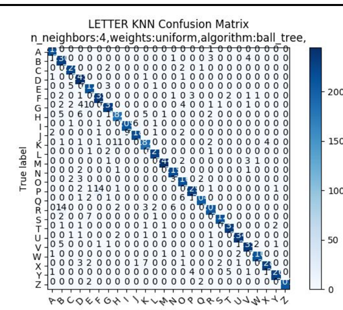
Best Parameters

Alg	Parameter	WIFI3	LETTER
DT	criterion	entropy	entropy
DT	splitter	best	best
DT	max_features	auto	auto
DT	max_depth	5	50
DT	min_samples_split	2	4
DT	min_samples_leaf	1	1
ADA	n_estimators	35	40
ANN	hidden_layer_sizes	10	90
ANN	activation	relu	relu
ANN	solver	adam	adam
ANN	learning_rate	constant	constant
ANN	learning_rate_init	0.010	0.010
ANN	max_iter	200	200
SVM	C	1	1
SVM	kernel	rbf	poly
SVM	degree	2	2
SVM	gamma	10	1
KNN	n_neighbors	7	4
KNN	weights	uniform	uniform
KNN	algorithm	ball_tree	ball_tree

Performance - sklearn.metrics.classification_report()

DataSet	Criteria	DT	ANN	ADA	SVM	KNN
wifi3	precision	0.94	0.98	0.96	0.97	0.98
wifi3	recall	0.94	0.98	0.95	0.97	0.98
wifi3	F1 score	0.94	0.98	0.95	0.97	0.98
letter	precision	0.84	0.95	0.84	0.94	0.95
letter	recall	0.84	0.95	0.84	0.94	0.95
letter	F1	0.84	0.95	0.83	0.94	0.95

Confusion Matrix

<table>
<tr><th></th><th>WiFi3</th><th>Letter</th></tr>
<tr><td>DT</td><td>
</td><td>
</td></tr>
<tr><td>ANN</td><td>
</td><td>
</td></tr>
<tr><td>ADA</td><td>
</td><td>
</td></tr>
<tr><td>SVM</td><td>
</td><td>
</td></tr>
<tr><td>KNN</td><td>
</td><td>
</td></tr>
</table>

Reading the Learning Curves [22]

Assuming an accuracy scorer (high is good):

- WiFi3, the DT algorithm training score decreases and plateau's indicating underfitting and high bias. The cross-validation score is increasing, so the algorithm is learning. The two lines converge, so there is low variance.
- WiFi3, the ANN algorithm is converging, indicating proper fitting and low variance.
- WiFI3 the SVM is overfitting. The validation curve plateau's indicating the algorithm is no longer learning from the additional data.
- WiFi3 the KNN algorithm validation curve actually start to decrease. This indicates the algorithms is no longer learning.
- Letter, the DT is overfitting. Possibly because the depth is too large in comparison to the split and leaf parameters.
- Letter, ADA is converging. There is a high variance. This could be solved by providing more data.
- Letter, the ANN, SVM, and KNN are all over fitting.

Analysing the Results - Lessons Learned

- Best parameters from Sklearn Gridsearch tend to overfit.
- Gridsearch provides statistics for all the test it ran. Providing valuable data to mine for information.
- ADA the decision tree must be simple to keep from overfitting.
- The DT algorithm is very sensitive the max_depth, min_samples_split, and min_samples_leaf hyperparameters. Tuning the DT algorithm consisted of trying to find the best combination of those three numeric parameters. If any of the hyperparameters are too high, DT tends to overfit.
- The ANN hidden layer size has a almost linear impact on learning capacity assuming the dataset is large enough. This is shown in the letter ANN validation curves. If there are too many nodes for the daraset, learning saturates. Saturation is shown in the WiFi ANN learning curves.
- The SVM is very sensitive to the kernel chosen. The datasets required different kernels as shown in the validation curves.
- The KNN algorithm is very sensitive to overfitting based on the validation data.

Were the Datasets Actually Interesting?

Both dataset turned out to be interesting from an analytical point of view. The WiFi dataset gave me a chance to analyze the effects of correlating attributes and random attributes on the algorithms. The letter dataset was very different from the wifi dataset giving me a chance to analyze the effect of different datasets on the algorithms.

References

1. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
2. https://archive.ics.uci.edu/ml/datasets/letter+recognition
3. http://archive.ics.uci.edu/ml/datasets/Wireless+Indoor+Localization
4. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
5. http://scikit-learn.org/0.16/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier
6. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
7. http://scikit-learn.org/0.16/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
8. http://scikit-learn.org/0.16/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier
9. https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html
10. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
11. http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
12. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html#sklearn.model_selection.learning_curve
13. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.validation_curve.html#sklearn.model_selection.validation_curve
14. "Deep Learning" (2016), Ian Goodfellow and Yoshua Bengio and Aaron Courville
15. http://scikit-learn.org/dev/modules/svm.html#kernel-functions
16. http://scikit-learn.org/stable/auto_examples/model_selection/plot_validation_curve.html#sphx-glr-auto-examples-model-selection-plot-validation-curve-py
17. http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py
18. http://scikit-learn.org/dev/modules/preprocessing.html#preprocessing
19. http://scikit-learn.org/stable/modules/model_evaluation.html
20. http://www.icmla-conference.org/icmla10/CFP_Tutorial_files/jose.pdf
21. http://www.springer.com/cda/content/document/cda_downloaddocument/9783319591612-c2.pdf?SGWID=0-0-45-1606592-p180867342
22. http://www.ritchieng.com/machinelearning-learning-curve/

23. Please see readme and source code for complete list of code origins