Comp 6721

Artificial Intelligence

Mini Project 2 Report

Author:	Student Id:
Raheleh Zarei Chamgordani	40059289

November 12, 2018

Department of Computer Science and Software Engineering

Table of Contents

1. Objective

In this project, the goal is to experiment three different algorithms, Decision Tree, Naïve Bayes and Multi Layer Perceptron classifier. We need to reveal what is the best performance of each algorithm from classification point of view and what is the best algorithm between three options for each dataset.

Accuracy, precision, recall and f-measure are metrics that are used to compare the algorithms in their different states and algorithms together.

2. Algorithms

In this project, we used three different algorithms. First algorithm is Decision Tree, which works based on choosing the features to classify new data. Second is Naïve bays that works base on probability of hypothesis considering the conditionally independent features. Third algorithm is SVM (Supported Vector Machin) that will be explained in more details in section 2.3

As the aim of this project, two different training dataset, validation set and test dataset will use to experiment the efficiency of each algorithms by changing hyper- parameters of each algorithm. We experiment these algorithms to realize which one and under which circumstances is the best algorithm for building a model to be able to obtain the better classification on new data.

2.1 Decision Tree

First algorithm is decision tree. We will use following parameters of this algorithm to obtain the value of each that lead to a better performance.

Min_samples_leaf: The minimum number of samples required to be at a leaf node.
Splitter: The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.
Max_feature: The number of features to consider when looking for the best split.
Max_depth: The maximum depth of the tree.
Presort: Whether to pre-sort the data to speed up the finding of best splits in fitting.

Min_samples_split: The minimum number of samples required to split an internal node.
(sklearn.tree.DecisionTreeClassifier)

2.1.1 Experiments with two datasets

First training dataset contains 1096 instance of data that are classified into 50 classes of uppercase and lowercase alphabets. Second training set is consist of 6400 samples that are classified into 9 classes. These training sets are used to design a model by using decision tree algorithm.
The created model used to classify the validation dataset. In this of this project, accuracy, precision, recall and f-measure are metrics that are used to helps us to choose the best combination of hyper-parameters that result higher efficiency. The created model will be used to classify the test set. The following information shows the results of decision tree by default values of the hyper-parameters to get a baseline idea of the performance.

Dataset 1		Dataset 2		
Accuracy	→0.29377	Accuracy	→	0.7625
Precision	→0.30145	Precision	→	0.69424
Recall	→0.28793	Recall	→	0.7175
F-measure	→0.28373	F-measure	→	0.70417

At first round of changing hyper-parameters, each parameter changed separately keeping the others as default. The reason for this is to see the impact of each selected parameter in model's efficiency.

The following tables is the summery of changing each parameter separately just by random values of each parameter for dataset1 (table 1) and dataset2 (table 2) to get an idea how they change the performance. We will see them in details later.

Metrics	Max depth=5	presort	Min sample split=20	Min samples leaf=10	Splitter = random	Max feature = 10	Max features = 500
Accuracy	0.14591	0.27042	0.26848	0.26459	0.28599	0.23540	0.291828
Precision	0.11083	0.27527	0.30667	0.29409	0.30227	0.2318081	0.285017
Recall	0.14086	0.26924	0.26757	0.26332	0.282793	0.232585	0.287547
F-measure	0.09673	0.26041	0.26062	0.25185	0.28170	0.226846	0.274670

Table 1

Metrics	Max depth=5	presort	Min sample split=20	Min samples leaf=10	Splitter = random	Max feature = 10	Max features = 500
Accuracy	0.596	0.7655	0.7705	0.772	0.7635	0.663	0.7445
Precision	0.58795	0.68863	0.7190	0.7071	0.6906	0.56955	0.6625
Recall	0.4439	0.7136	0.71529	0.7141	0.7116	0.59279	0.6752
F-measure	0.44133	0.69928	0.71489	0.7092	0.6997	0.57931	0.6663

Table 2

Changing maximum depth of the tree from “None”, which is default, and it means that nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split, to depth of 5, decrease all the metrics significantly for both dataset. This shows that limiting the max depth of tree prevents model to expand all nodes and address each sample to its appropriate class, thus it reduce the values of all metrics.

Changing presort parameter from “False” which is default value to “True” reduce all metrics by ≈ 0.1 for both datasets.

By increasing the value of Min_sample_split, metrics values decrease. Default value is 2 and for large values, this parameter decrease efficiency remarkably in dataset1 and with less impact in database2. The metrics values in table are when this parameter is set to 20.

By increasing the value of Min_sample_leaf, metrics values decrease. Default value is 1 and for large values, this parameter decrease efficiency significantly in dataset1, however in dataset2 (as figure 6 illustrates in details) accuracy of validation set is higher than training set for %20 (I cant get a conclusion of this behaviour) . The metrics values in tables are when this parameter is set to 20.

Changing splitter parameter from its default, which is “best” to “random” decrease the metrics values as it tries to choose the best splitter among the random possible split. It doesn’t have a great impact in both datasets.

Max_feature is the value that decide how many features considered to be used as splitter. Reducing the number of available features to split will reduce the metrics values. Tables illustrate the metrics values for number of 10 and 500 as per of this value. Default value is the entire available features in training set.

Now, we want to reach the high efficiency of our decision tree algorithms for both dataset. Therefore, we need to know, what combination of hyper-parameters gives us highest performance. In order to achieve this goal at first we need to understand what is the best value of each parameter individually and then their combination.

For this happen, we need to look at the plots of accuracy for continues parameters. These plots shows us how the changing parameters that take a range of value influence the performance.

According the plots below, increasing the Max_depth from zero to 50 leads to overfitting for both datasets. The tree completely predicts all of the training dataset; however, it can not generalize the findings for new data. Best Max_depth is around 22 for dataset 1 (figure1) and around 10 for dataset 2 (figure 2).

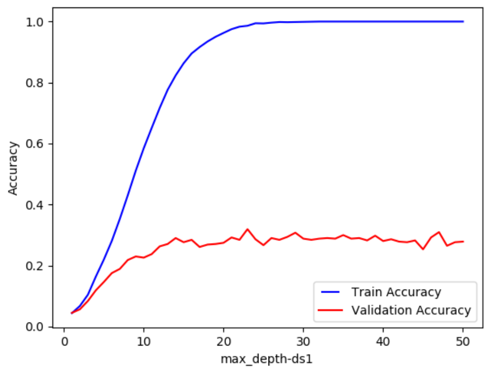


Figure 1

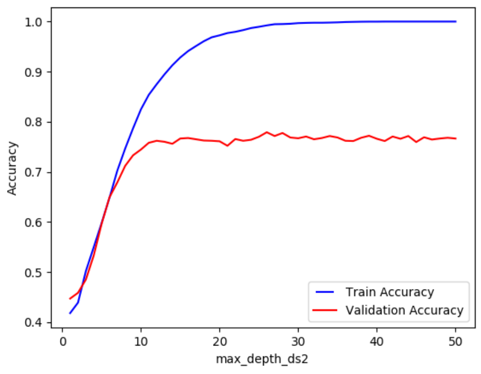


Figure 2

By using the same interpretation, following plots reveal the influence of increasing number of samples at each node for dataset1 (figure 3) and dataset2 (figure 4).

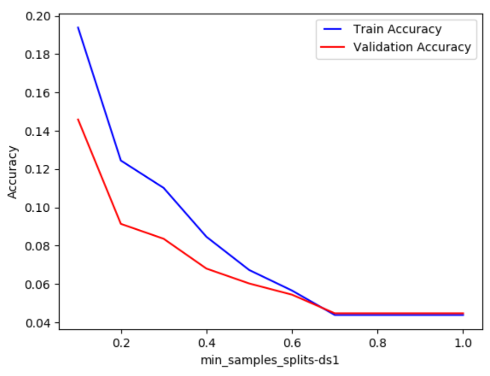


Figure 3

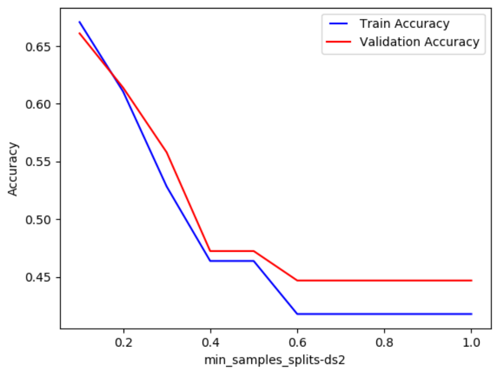


Figure 4

This is an underfitting case because when we consider 100% of the instances at each node, the model cannot learn enough about the data.

The same experience is for Min_samples_leaf. Increasing value of this parameter may cause underfitting for both dataset1 (figure 5) and dataset2 (figure 6). Choosing small number of samples in the leaves lead to a better result.

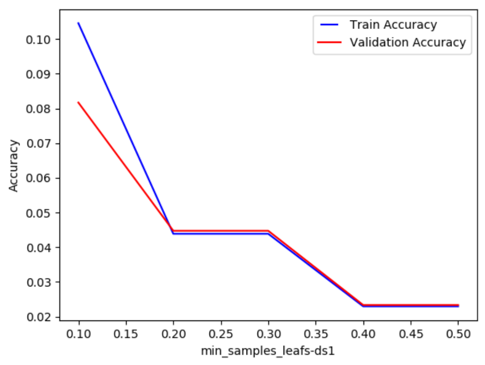


Figure 5

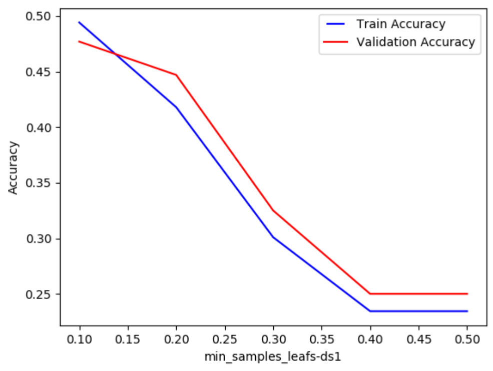


Figure 6

Experiment with changing Max_feature parameter is also interesting. This case leads to an overfitting too. It is unexpected to get overfitting for all values of max_features. However, according to sklearn documentation for decision tree, the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features. (Fraj)

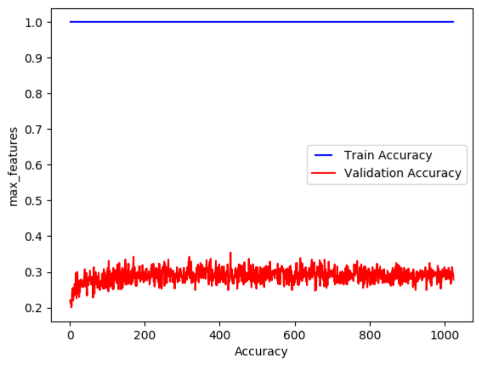


Figure 7

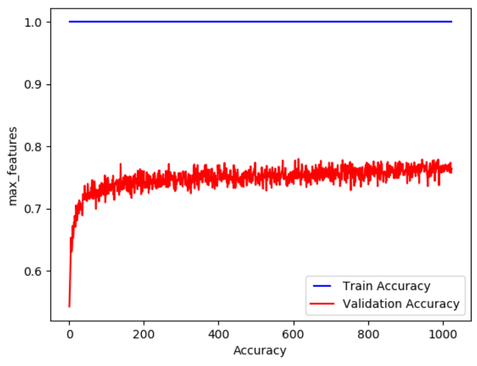


Figure 8

2.2 Naïve Bayes

Second algorithm used for this project, is naïve bayes. We use following hyper-parameter to experiment the dataset1 and dataset2 and recognize which condition gives better performance.

Alpha: Additive smoothing parameter
fit_prior: Boolean, optional (default=True) whether to learn class prior probabilities or not. If false, a uniform prior will be used.

Following information shows the metrics values for dataset1 and dataset2 with default parameters values.

2.2.1 Experiments with two datasets

Dataset 1			Dataset 2		
Accuracy	→	0.59922	Accuracy	→	0.8005
Precision	→	0.61296	Precision	→	0.7462
Recall	→	0.5957	Recall	→	0.7783
F-measure	→	0.5923	F-measure	→	0.7566

In table below the parameter, fit_prior changed to False and result has been presented for dataset1 (figure 9) and daset2 (figure 10). Changing this parameter has not a significant change in dataset1 and dataset2, which can be a sign of tuned data, and each class has pretty the same probability.

Metrics	Fit_prior=False
Accuracy	0.59727
Precision	0.61141
Recall	0.59391
F-measure	0.59029

Figure 9

Metrics	Fit_prior=False
Accuracy	0.799
Precision	0.7417
Recall	0.7821
F-measure	0.7545

Figure 10

Changing alpha as a continues parameter is shown in figures below.

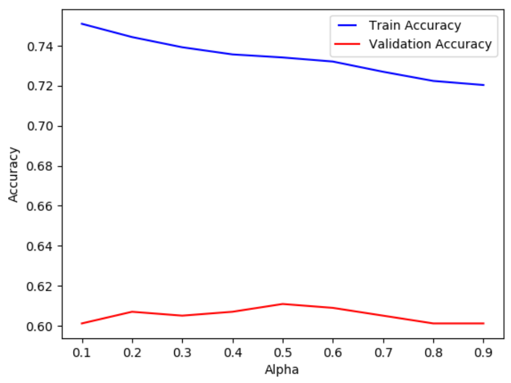


Figure 11

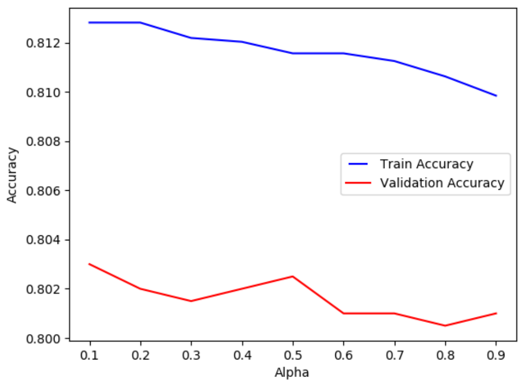


Figure 12

Figure 11, the plot for dataset1 shows that best accuracy is when the alpha is set to 0.5 for validation set. This is the same fact for dataset2.

2.3 Multi layer perceptron classifier

Class MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using backpropagation.

MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating point feature vectors; and array y of size

(n_samples, class labels), which holds the target values (class labels) for the training samples.
MLP trains using Backpropagation. More precisely, it trains using some form of gradient descent and the gradients are calculated using Backpropagation. For classification, it minimizes the Cross-Entropy loss function, giving a vector of probability estimates P (Y|X) per sample x. MLPClassifier supports multi-class classification by applying Softmax as the output function. (Neural network models (supervised), n.d.)

2.3.1 Experiments with two datasets

Same metrics as decision tree and naïve Bayes, used for this algorithm. Following information shows the metrics values for dataset1 and dataset2.

Dataset 1			Dataset 2		
Accuracy	→	0.612840	Accuracy	→	0.895
Precision	→	0.625344	Precision	→	0.8666
Recall	→	0.611942	Recall	→	0.8671
F-measure	→	0.607622	F-measure	→	0.86602

Following hyper-parameter are used to experiment MLP classifier in different situation and obtain the most efficient combination of these parameters.

hidden_layer_sizes: Number of hidden layers and number of neurons in each layer
Activation: Activation function for hidden layer. Available options are {‘identity’, ‘logistic’, ‘tanh’, ‘relu’}. Default is ‘relu’
We will experiment with following activation functions.
Logistic, the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
Tanh, the hyperbolic tan function, returns f(x) = tanh(x).
Relu, the rectified linear unit function, returns f(x) = max (0, x)
Solver: solver for weight optimization. Available options are {‘lbfgs’, ‘sgd’, ‘adam’}. Default is ‘adam’
lbfgs is an optimizer in the family of quasi-Newton methods.
sgd refers to stochastic gradient descent.
adam refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

The following tables is the summery of changing each parameter separately for dataset1 (table 8) and dataset2 (table 9) to get an idea how they change the performance.

First parameter is hidden layer that comes in a tuple (number of layers, number of neurons). The figures below reveal how this parameter influences the accuracy of designed model.

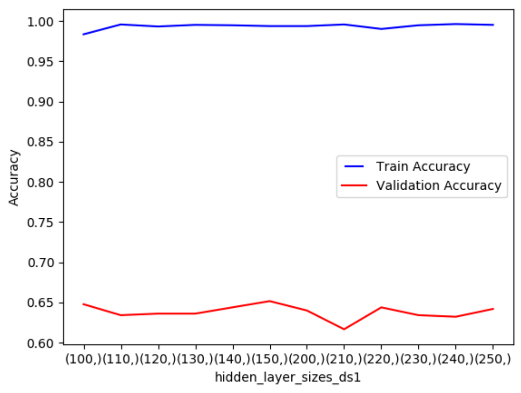


Figure 13

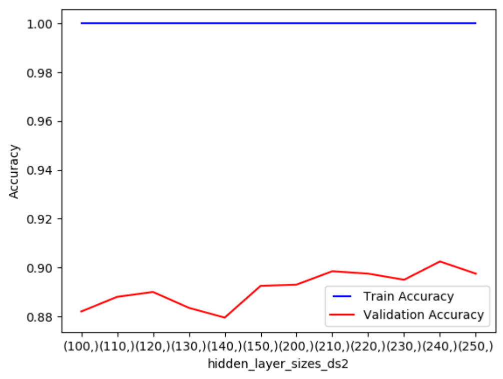


Figure 14

Another important parameter is activation function that is selectable between 4 different options. Figure 9 and 10 are for dataset and dataset2 respectively. For both dataset, 'logistic' activation function gives better higher accuracy. Accuracy for training set in dataset1 has the same value for all the activation function. This is a case of overfitting.

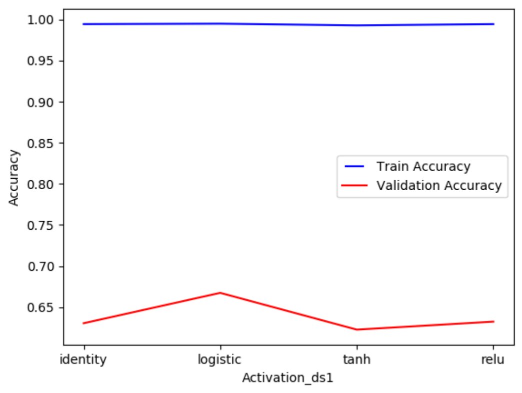


Figure 15

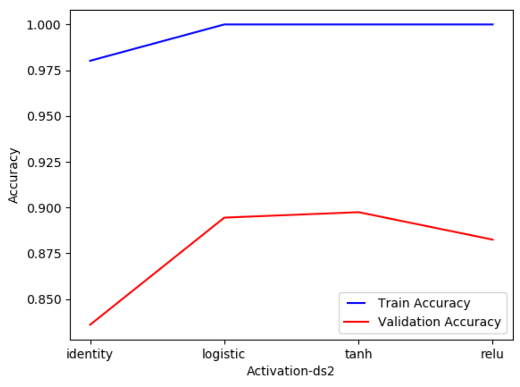


Figure 16

Solver is a hyper-parameter that might influence in model performance. The two following figures shows the effect of this hyper-parameter for dataset1 (figure15) and dataset2 (figure 16). For dataset1 solvers, ‘lbfgs’ and ‘adam’ work better than sgd for both training and validation set. While for dataset2, accuracy of validation is highest for ‘adam’ first and ‘sgd’ is in second place.

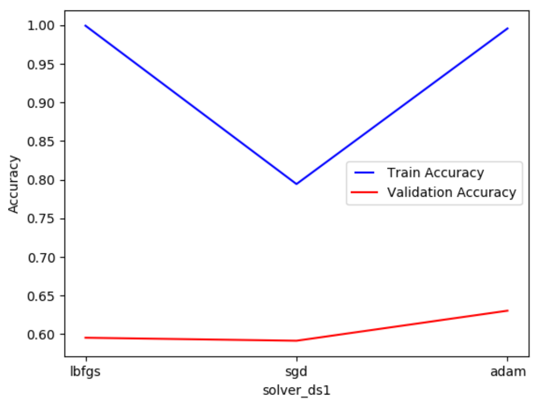


Figure 17

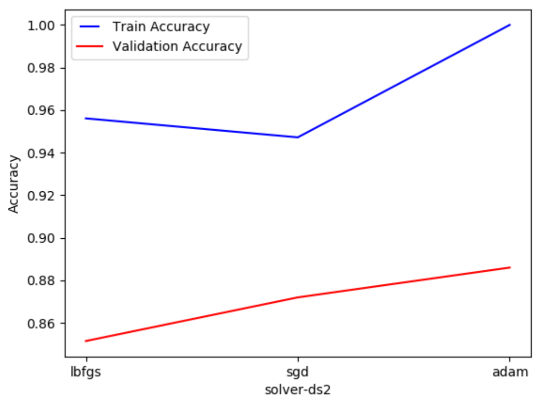


Figure 18

3. Best parameter’s values

For dataset1, Decision Tree algorithm executed for validation dataset with following parameters and values that assumed to give most efficient result.

max_features=500, min_samples_split=2, min_samples_leaf=1, max_depth=20

Accuracy →0.31128, Precision→0.322, Recall→0.306, F-measure→0.300

For dataset2, decision tree algorithm executed for validation dataset with following parameters and values that assumed to give most efficient result.

max_features=1024, min_samples_split=2, min_samples_leaf=1, max_depth=16

Accuracy →0.765, Precision→0.6994034, Recall→0.7028000, F-measure→0.698

For dataset1, Naïve Bayes algorithm executed for validation dataset with following parameter and value that assumed to give most efficient result.

Alpha=0.5

Accuracy→0.6108, Precision→0.6255, Recall→0.60822, F-measure→0.6042

For dataset2, Naïve Bayes algorithm executed for validation dataset with following parameter and value that assumed to give most efficient result.

Alpha=0.5

Accuracy→0.8025, Precision→0.7525, Recall→0.7797, F-measure→0.7608

For dataset1, Multi Layer Perceptron algorithm executed for validation dataset with following parameters and values that assumed to give most efficient result.

hidden_layer_sizes= (150,), activation='logistic', solver='adam'

Accuracy→0.67315, Precision→0.6765, Recall→0.669, F-measure→0.66194

For dataset2, Multi Layer Perceptron algorithm executed for validation dataset with following parameters and values that assumed to give most efficient result.

hidden_layer_sizes= (240,), activation='logistic', solver='adam'

Accuracy→0.901, Precision→0.877, Recall→0.8724, F-measure→0.873

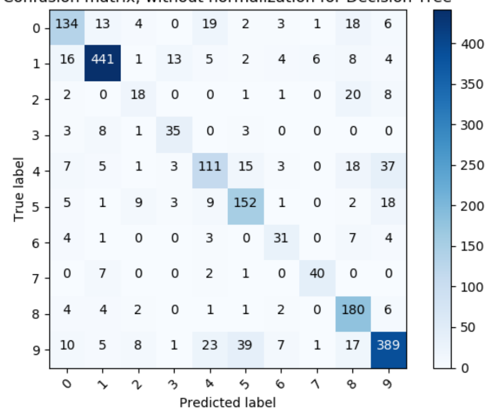
4. Conclusion

Regarding the result from comparing different algorithms, we go with MLP classifier algorithm for test sets however; the executions speed of this algorithm is high in comparison with Naïve Bayes and decision tree. (Confusion matrixes has been shown for dataset2 from three algorithms)

According to the experiments, results of metric values are better for dataset2 as there are more instance in training dataset, so our model will be able to have a better generalization. Results where in accord with our expectation regarding the compression of results for two datasets and each algorithm separately for datasets.

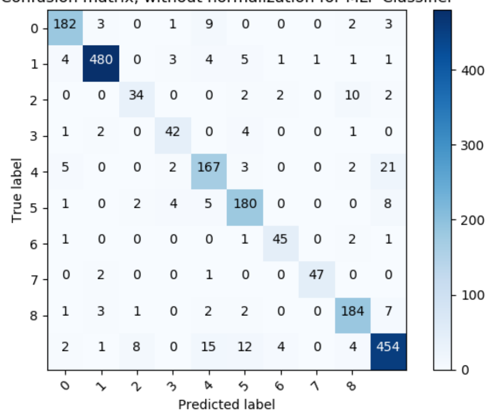
If I had more time I would like work on precision- recall curve for each mode on each data set which is a more involved method to tune our model by searching for the best hyper-parameters and keeping the classifier with the highest recall and precision value.

Confusion matrix, without normalization for Decision Tree


True label
Predicted label
400
350
300
250
200
150
100
50
0

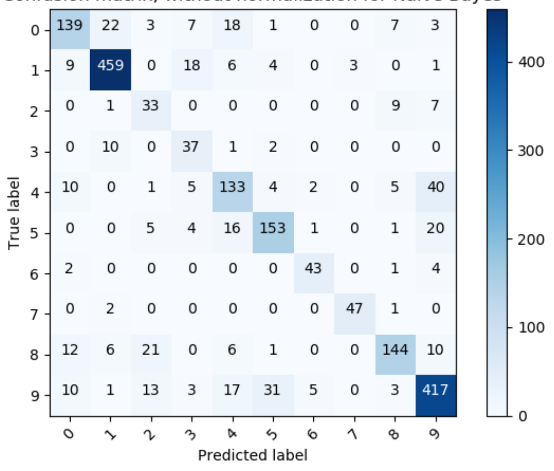
	0	1	2	3	4	5	6	7	8	9
0	134	13	4	0	19	2	3	1	18	6
1	16	441	1	13	5	2	4	6	8	4
2	2	0	18	0	0	1	1	0	20	8
3	3	8	1	35	0	3	0	0	0	0
4	7	5	1	3	111	15	3	0	18	37
5	5	1	9	3	9	152	1	0	2	18
6	4	1	0	0	3	0	31	0	7	4
7	0	7	0	0	2	1	0	40	0	0
8	4	4	2	0	1	1	2	0	180	6
9	10	5	8	1	23	39	7	1	17	389

Confusion matrix, without normalization for MLP Classifier


True label
Predicted label
400
300
200
100
0

	0	1	2	3	4	5	6	7	8	
0	182	3	0	1	9	0	0	0	2	3
1	4	480	0	3	4	5	1	1	1	1
2	0	0	34	0	0	2	2	0	10	2
3	1	2	0	42	0	4	0	0	1	0
4	5	0	0	2	167	3	0	0	2	21
5	1	0	2	4	5	180	0	0	0	8
6	1	0	0	0	0	1	45	0	2	1
7	0	2	0	0	1	0	0	47	0	0
8	1	3	1	0	2	2	0	0	184	7
	2	1	8	0	15	12	4	0	4	454

Confusion matrix, without normalization for Naive Bayes


True label
Predicted label
400
300
200
100
0

	0	1	2	3	4	5	6	7	8	9
0	139	22	3	7	18	1	0	0	7	3
1	9	459	0	18	6	4	0	3	0	1
2	0	1	33	0	0	0	0	0	9	7
3	0	10	0	37	1	2	0	0	0	0
4	10	0	1	5	133	4	2	0	5	40
5	0	0	5	4	16	153	1	0	1	20
6	2	0	0	0	0	0	43	0	1	4
7	0	2	0	0	0	0	0	47	1	0
8	12	6	21	0	6	1	0	0	144	10
9	10	1	13	3	17	31	5	0	3	417

References

Fraj, M. B. (n.d.). *Parameter tuning for Decision Tree.* Retrieved from https://medium.com/@.

Neural network models (supervised). (n.d.). Retrieved from https://scikit-learn.org.

sklearn.tree.DecisionTreeClassifier. (n.d.). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html