Project Report for Intro to AI (CS 520)

Image Classification

Submitted to Rutgers, the State University of New Jersey

Towards the partial fulfillment for the Award of the Degree of

MASTERS OF SCIENCES

In Computer and Information Sciences

2022-2023

By

Niyati Jain - 216008777 (nsj39)

Agrani Swarnkar - 219002416 (as4154)

Dhruv Patel - 219000059 (dp1224)

Under the Guidance of

Mr. Abdeslam Boularias

Department of Computer Science Engineering

School of Arts and Sciences

Rutgers University – New Brunswick

New Jersey

Table of Contents

Abstract

This project is an image classifier program that uses 3 different kinds of machine learning algorithms to classify images that are stored in text files. Currently it can classify either single digit hand written numbers, or determine whether or not there is a face in the image. The three algorithms used are: - Naive Bayes classifier - Single layer multiclass perceptron - Support Vector Machine with linear kernel.
The implementation of the primary two algorithms is done from scratch using Numpy. SVM is implemented using the scikit learn library function. The accuracy on the Naive Bayes algorithm is approximately 71% for digits and 90.7% for faces. The perceptron is able to achieve 82.4% accuracy for digits, and 90.7% for faces. SVM performs the best out of the three algorithms for classifying digits with 87.5% accuracy, and is able to achieve 90.3% accuracy on faces. Though, these classifiers provide an accurate reading based on the data available, the accuracy can be substantially improved by implementing filtering or other alternatives of feature extraction.

Introduction

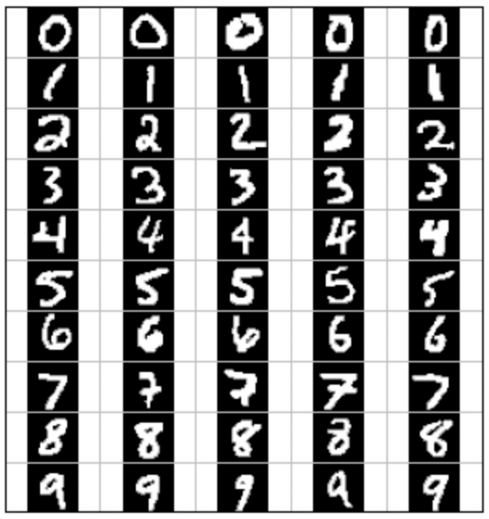
Optical Character Recognition or OCR provides a variety of alternatives for reading, locating, and identifying text in photos and labels. Whenever we think of optical character recognition, a lot of paper comes to mind.

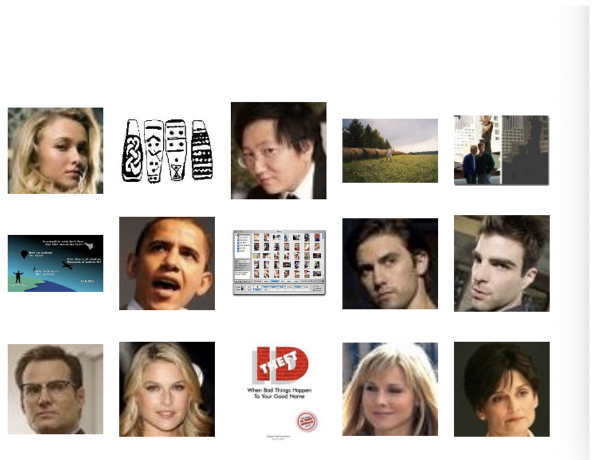
In addition to taking up a lot of storage space, papers, including legal and secret personal documents, can also be problematic if they are misplaced. OCR steps in at this point and plays a crucial role in the digitalization of documents. A class of computer vision issues called OCR Machine Learning transforms handwritten or typewritten text from a digital image into text that computers can understand. The output is subsequently processed, saved, and edited by your system as part of data entry software or as a text file.

People have tried a number of conventional methods to address the OCR Machine Learning challenge via computer vision since the rise of machine learning text recognition in 2014. However, in order for businesses to deploy their machine learning applications at scale, it is critical to investigate new techniques to make our models robust to these fluctuations.

Face recognition is a biometric identification method that uses the individual's face's distinctive features to identify them. By comparing the face print to a database of recognized faces, the majority of facial recognition systems operate. The system can recognize the person if there is a match. However, the system cannot identify the person if the face print is not stored in the database.

The identification of criminals is one security use for facial recognition technology.


Which Digit?


Which are Faces?

Perceptron

The Perceptron algorithm is a single-layer neural network, which consists of a neuron that takes as input a row of data and predicts a class label. The class label can be computed using the weighted sum of input(Activation) and the bias which is set as 1. Activation can be calculated as:

Activation = Weights * Inputs + Bias

Output : Prediction = 1; If Activation > 0.0

Output : Prediction = 0; If Activation <= 0.0

It is always best practice to normalize the data before the application of the model as the inputs are multiplied by the model coefficients(input weights). The input weights are trained using the stochastic gradient descent optimization algorithm.

The perceptron works by feeding some examples from the training dataset to the model one at a time. First, the model makes a prediction, and then the error is computed. To reduce the errors, the weights of the model can be updated. This process is termed as the Perceptron update rule and is repeated for all examples of the dataset(epochs). With the small proportion of the errors found in each batch, we can update the weights of our model. The proportion is determined by a hyperparameter called the learning rate, which is set to a small value so that learning cannot happen immediately which would rather result in a lower skill model, called premature convergence of the optimization (search) procedure for the model weights.

weight(t + 1) = weight(t) + learningRate * (expected(i) – predicted(i)) * input(i)

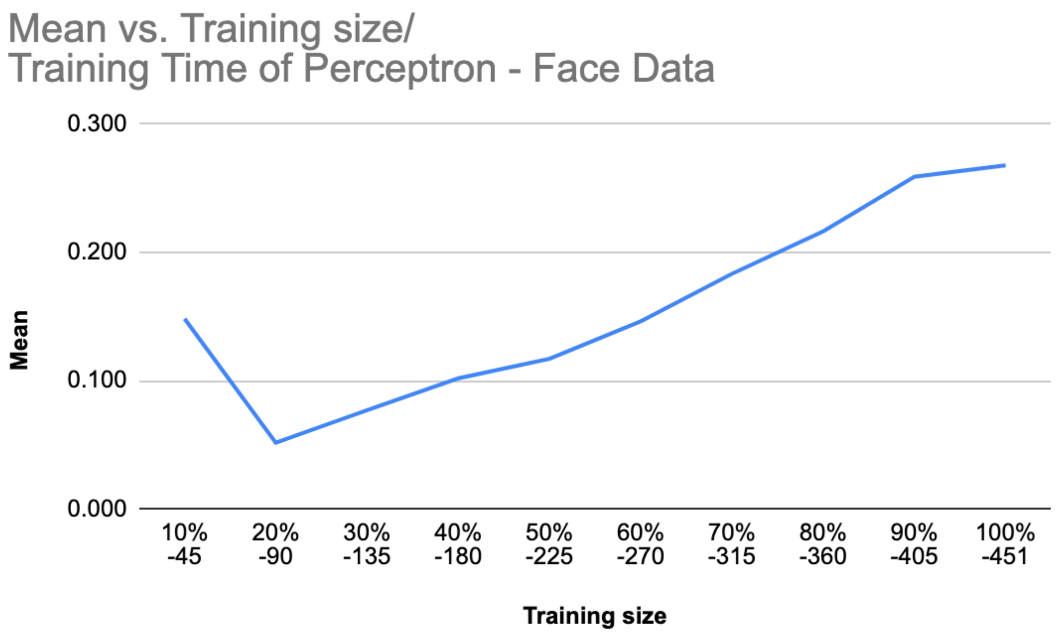
When the model stops improving or when the maximum number of epochs is performed, the training is concluded. The model weights are initially set to small random values. Furthermore, we shuffle the training dataset before each training epoch to accelerate and improve the training process of our model. We might get different results as outputs each time the algorithm is run as it is stochastic. Hence, we should summarize the performance of the algorithm by computing the mean classification accuracy of the repeated evaluations of the datasets.

The algorithm's learning rate (alpha = 0.05) and number of iterations(iterations = 10) are hyperparameters that can be set help in tuning the model for maximum accuracy..

Face Data:

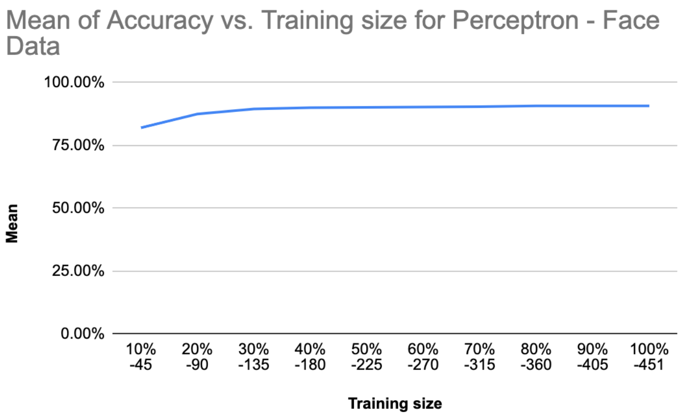
Training Time

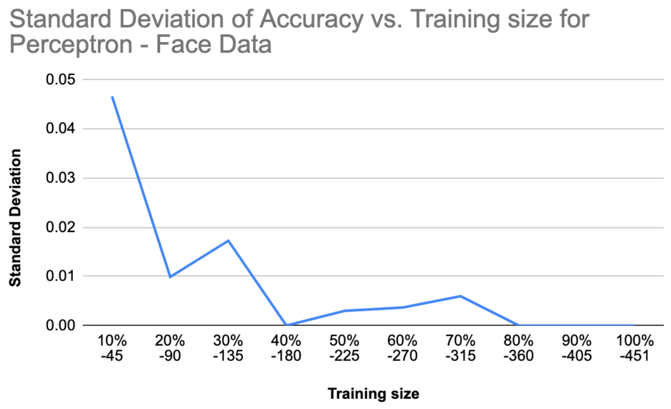
Training size/ Training Time(in s) for Naive Bayes - Face Data	10% -45	20% -90	30% -135	40% -180	50% -225	60% -270	70% -315	80% -360	90% -405	100% -451
Iteration 1	0.0117	0.0219	0.0318	0.0419	0.0525	0.0628	0.0731	0.0819	0.0916	0.1019
Iteration 2	0.0115	0.0216	0.0315	0.0416	0.0523	0.0629	0.0731	0.0817	0.0916	0.1019
Iteration 3	0.0115	0.0216	0.0315	0.0416	0.0523	0.0627	0.0731	0.0817	0.0917	0.1018
Iteration 4	0.0115	0.0216	0.0315	0.0416	0.0523	0.0626	0.0722	0.0816	0.0919	0.1019
Iteration 5	0.0115	0.0216	0.0315	0.0423	0.0525	0.0627	0.0716	0.0816	0.0917	0.1018
Mean	0.0115	0.0217	0.0315	0.0418	0.0524	0.0628	0.0726	0.0817	0.0917	0.1019
Standard Deviation	0.0001	0.0001	0.0001	0.0003	0.0001	0.0001	0.0007	0.0001	0.0001	0.0001



Accuracy

Training size/ Accuracy of Perceptron - Face	10% -45	20% -90	30% -135	40% -180	50% -225	60% -270	70% -315	80% -360	90% -405	100% -451
Iteration 1	74.7%	86.0%	87.3%	90.0%	90.0%	90.7%	89.3%	90.7%	90.7%	90.7%
Iteration 2	80.0%	87.3%	91.3%	90.0%	90.0%	90.0%	90.7%	90.7%	90.7%	90.7%
Iteration 3	85.3%	87.3%	88.0%	90.0%	90.7%	90.0%	90.7%	90.7%	90.7%	90.7%
Iteration 4	84.7%	88.0%	90.0%	90.0%	90.0%	90.0%	90.7%	90.7%	90.7%	90.7%
Iteration 5	85.3%	88.7%	90.7%	90.0%	90.0%	90.7%	90.7%	90.7%	90.7%	90.7%
Mean	82.0%	87.5%	89.5%	90.0%	90.1%	90.3%	90.4%	90.7%	90.7%	90.7%
Standard Deviation	0.0467	0.0099	0.0173	0.0000	0.0030	0.0037	0.0060	0.0000	0.0000	0.0000

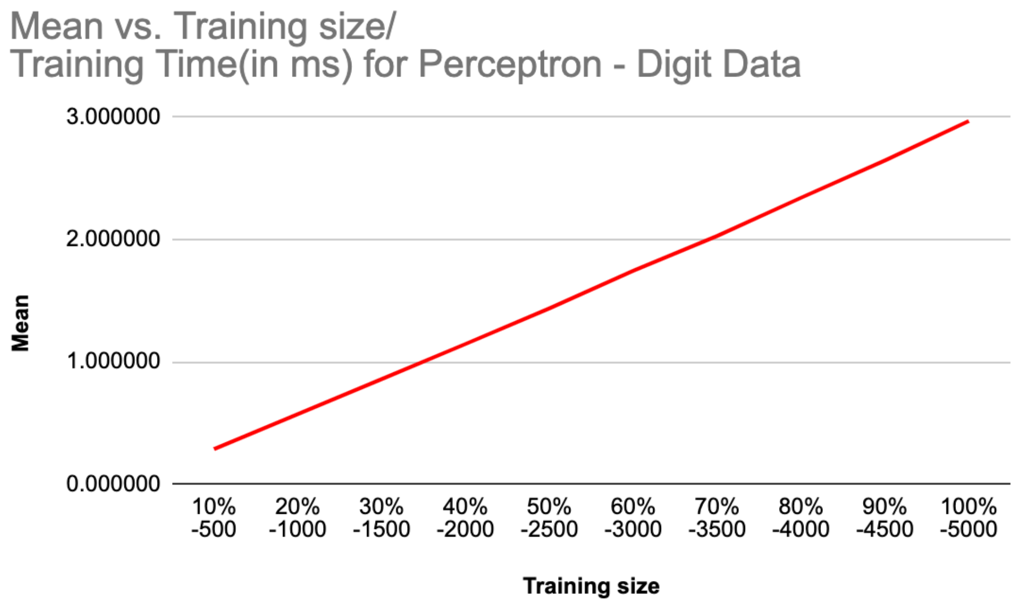




Digit Data:

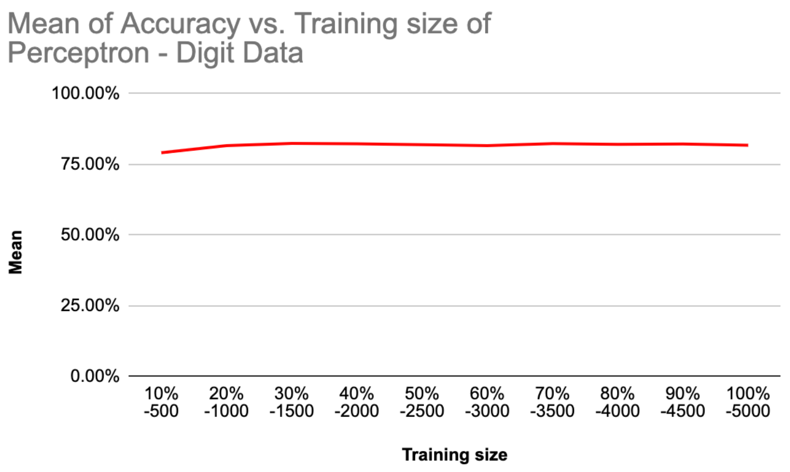
Training Time

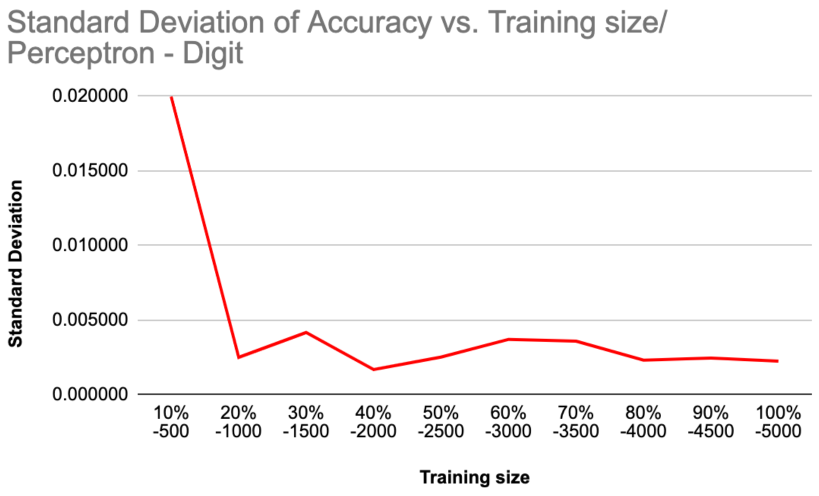
Training size/ Training Time(in s) for Perceptron - Digit Data	10% -500	20% -1000	30% -1500	40% -2000	50% -2500	60% -3000	70% -3500	80% -4000	90% -4500	100% -5000
Iteration 1	0.2861	0.5672	0.8582	1.1428	1.4357	1.7391	2.0110	2.3481	2.6294	2.9629
Iteration 2	0.2857	0.5860	0.8631	1.1561	1.4435	1.7493	2.0261	2.3424	2.6383	2.9335
Iteration 3	0.2865	0.5753	0.8562	1.1427	1.4375	1.7416	2.0313	2.3415	2.6594	2.9202
Iteration 4	0.2865	0.5653	0.8569	1.1417	1.4389	1.7513	2.0347	2.3469	2.6438	2.9429
Iteration 5	0.2858	0.5673	0.8608	1.1474	1.4337	1.7464	2.0439	2.3297	2.6623	3.0841
Mean	0.2861	0.5722	0.8590	1.1462	1.4379	1.7455	2.0294	2.3417	2.6466	2.9687
Standard Deviation	0.0004	0.0086	0.0029	0.0060	0.0037	0.0051	0.0122	0.0073	0.0140	0.0664



Accuracy

Training size/ Perceptron - Digit	10% -500	20% -1000	30% -1500	40% -2000	50% -2500	60% -3000	70% -3500	80% -4000	90% -4500	100% -5000
Iteration 1	74.4%	80.4%	79.1%	80.7%	81.5%	80.1%	80.6%	80.7%	81.3%	80.9%
Iteration 2	77.0%	80.4%	79.0%	81.7%	80.5%	80.3%	81.0%	81.4%	80.7%	80.9%
Iteration 3	77.3%	80.0%	79.1%	81.5%	80.4%	80.2%	81.2%	81.5%	80.5%	80.9%
Iteration 4	77.5%	80.3%	79.5%	81.1%	80.5%	79.7%	81.1%	81.6%	80.7%	80.6%
Iteration 5	77.9%	80.6%	79.4%	80.9%	80.3%	79.8%	81.0%	81.3%	80.5%	80.5%
Mean	76.8%	80.3%	79.2%	81.2%	80.6%	80.0%	81.0%	81.3%	80.7%	80.8%
Standard Deviation	1.4%	0.2%	0.2%	0.4%	0.5%	0.3%	0.2%	0.4%	0.3%	0.2%





Naive Bayes

In machine learning, if we are given a data(d) we are generally concerned about selecting the best hypothesis (h).Our hypothesis (h) may be the class to assign for a new data instance(d) in a classification problem.

Using Bayes Theorem, we can calculate the probability of a hypothesis given our prior knowledge that is the data(d). Bayes Rule is stated as follows:

P(h|d) = (P(d|h) * P(h)) / P(d)

Here,

- P(h|d) : posterior probability, i.e., probability of h given the d
- P(d|h) : probability of d given that the h is true.
- P(h) : prior probability, i.e., probability of h being true (not depending on the data).
- P(d) : probability of the data (not depending on the hypothesis).

From the posterior probability of different hypotheses, we select the one with the highest probability. This maximum probable hypothesis is called the maximum a posteriori (MAP) hypothesis and can be written as:

MAP(h) = max(P(h|d))

Substituting the value of **P(h|d)**, we get

MAP(h) = max((P(d|h) * P(h)) / P(d))

P(d) is a constant term and is only used to normalise. We can ignore it when we are calculating the most probable hypothesis:

MAP(h) = max(P(d|h) * P(h))

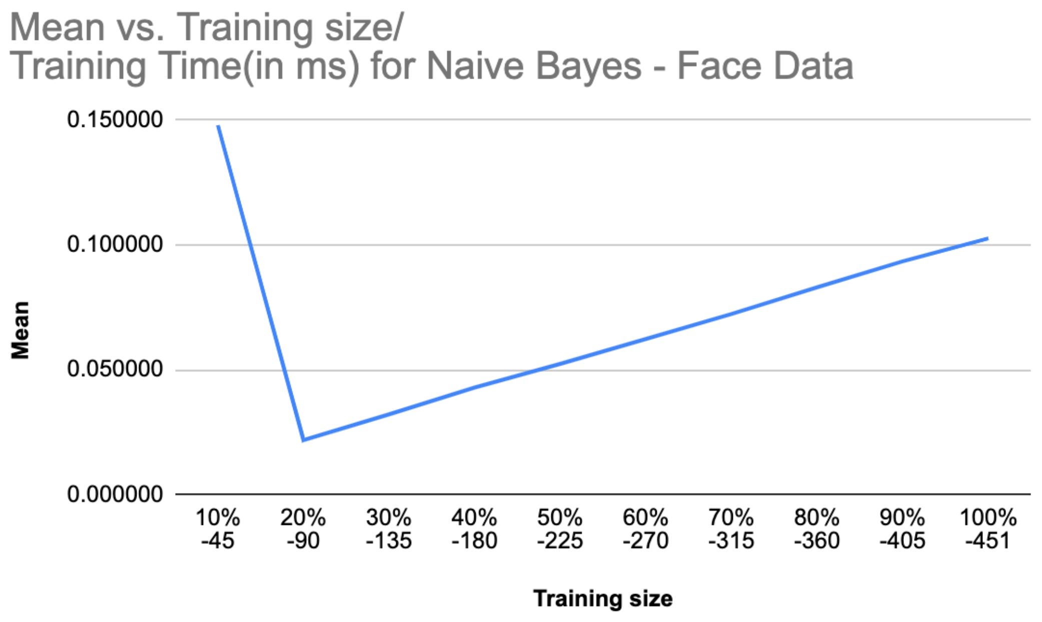
Naive Bayes is used for binary and multi-class classification problems. The simplest way to understand this technique is when it is described using binary or categorical input values.

The calculation of the probabilities for each hypothesis is simplified to make the calculation tractable. That is why the algorithm is termed as ‘Naive’ or Idiot Bayes. The values of each attribute P(d1,d2,d3|h) is assumed to be conditionally independent and calculated as P(d1|h)*P(d2|h) rather than being calculated which is a very strong assumption given the fact that attributes don’t interact. Still the performance is very well on data where this assumption does not hold.

Face Data:

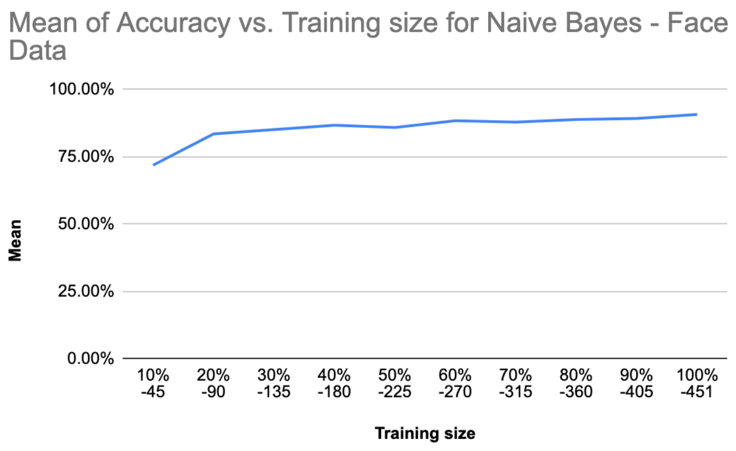
Training Time

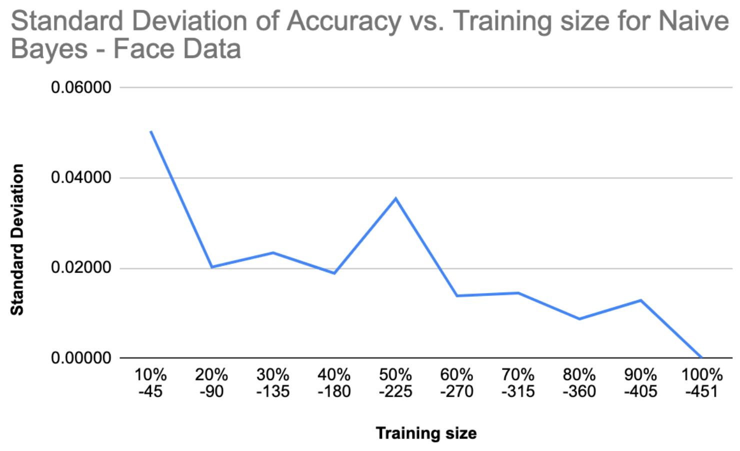
Training size/ Training Time(in s) for Naive Bayes - Face Data	10% -45	20% -90	30% -135	40% -180	50% -225	60% -270	70% -315	80% -360	90% -405	100% -451
Iteration 1	0.0117	0.0218	0.0320	0.0424	0.0518	0.0647	0.0725	0.0822	0.0935	0.1039
Iteration 2	0.0116	0.0217	0.0322	0.0431	0.0538	0.0618	0.0724	0.0832	0.0936	0.1033
Iteration 3	0.0117	0.0226	0.0323	0.0445	0.0517	0.0618	0.0733	0.0833	0.0935	0.1020
Iteration 4	0.0116	0.0216	0.0321	0.0428	0.0517	0.0617	0.0717	0.0833	0.0931	0.1020
Iteration 5	0.0117	0.0217	0.0322	0.0417	0.0528	0.0618	0.0717	0.0833	0.0936	0.1020
Mean	0.1479	0.0219	0.0322	0.0429	0.0524	0.0623	0.0723	0.0831	0.0935	0.1026
Standard Deviation	0.1443	0.0004	0.0001	0.0010	0.0009	0.0013	0.0007	0.0005	0.0002	0.0009



Accuracy

Training size/ Accuracy of Naive Bayes - Face	10% -45	20% -90	30% -135	40% -180	50% -225	60% -270	70% -315	80% -360	90% -405	100% -451
Iteration 1	64.7%	83.3%	82.0%	88.7%	82.7%	87.3%	88.0%	88.7%	90.0%	90.7%
Iteration 2	76.7%	86.0%	88.0%	85.3%	84.7%	90.7%	87.3%	89.3%	89.3%	90.7%
Iteration 3	75.3%	80.7%	84.0%	86.0%	83.3%	88.7%	88.0%	87.3%	88.7%	90.7%
Iteration 4	74.0%	84.7%	86.7%	84.7%	87.3%	88.0%	86.0%	89.3%	90.7%	90.7%
Iteration 5	68.7%	82.7%	84.7%	88.7%	91.3%	87.3%	90.0%	89.3%	87.3%	90.7%
Mean	71.9%	83.5%	85.1%	86.7%	85.9%	88.4%	87.9%	88.8%	89.2%	90.7%
Standard Deviation	0.0504	0.0202	0.0234	0.0189	0.0354	0.0138	0.0145	0.0087	0.0128	0.0000

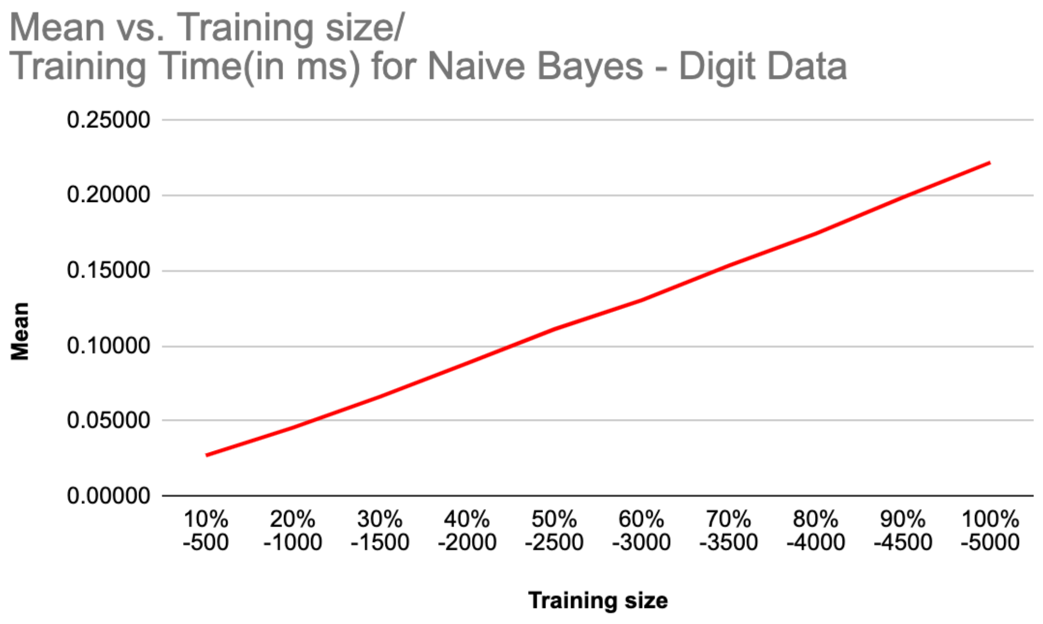




Digit Data:

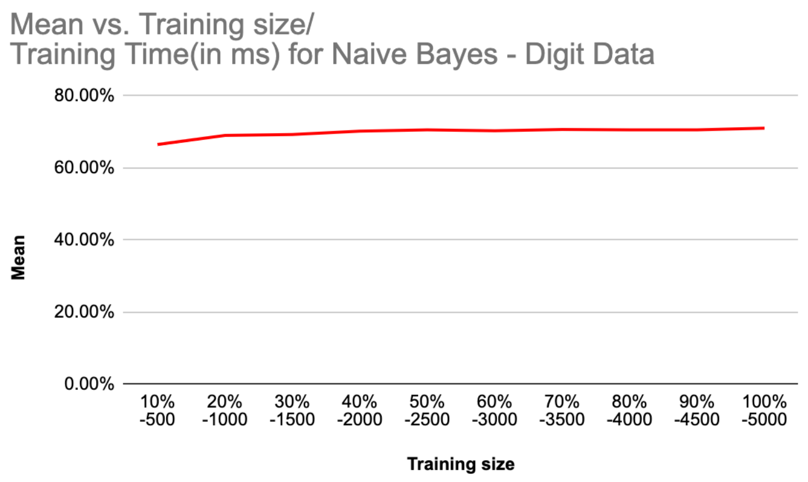
Training Time

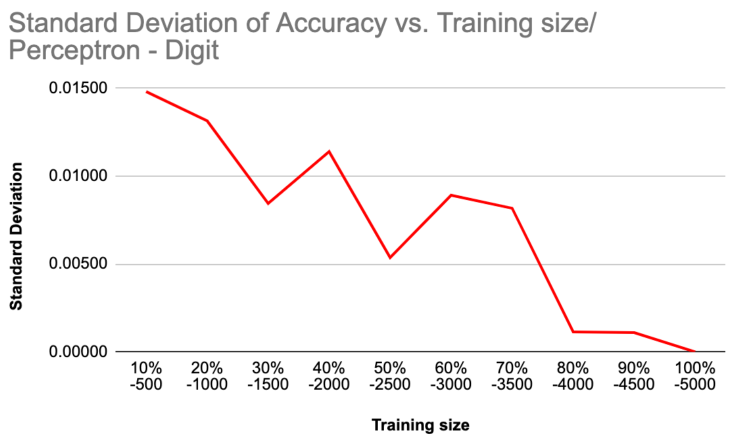
Training size/ Training Time(in s) for Naive Bayes - Digit Data	10% -500	20% -1000	30% -1500	40% -2000	50% -2500	60% -3000	70% -3500	80% -4000	90% -4500	100% -5000
Iteration 1	0.0231	0.0459	0.0663	0.0886	0.1122	0.1307	0.1553	0.1744	0.1983	0.2234
Iteration 2	0.0229	0.0460	0.0674	0.0885	0.1106	0.1298	0.1550	0.1759	0.1981	0.2242
Iteration 3	0.0407	0.0471	0.0662	0.0886	0.1109	0.1297	0.1525	0.1745	0.2003	0.2223
Iteration 4	0.0247	0.0446	0.0656	0.0885	0.1111	0.1296	0.1524	0.1743	0.1977	0.2213
Iteration 5	0.0240	0.0445	0.0658	0.0887	0.1111	0.1324	0.1521	0.1747	0.2004	0.2191
Mean	0.0271	0.0456	0.0663	0.0886	0.1112	0.1305	0.1535	0.1748	0.1989	0.2221
Standard Deviation	0.0077	0.0011	0.0007	0.0001	0.0006	0.0012	0.0016	0.0006	0.0013	0.0020



Accuracy

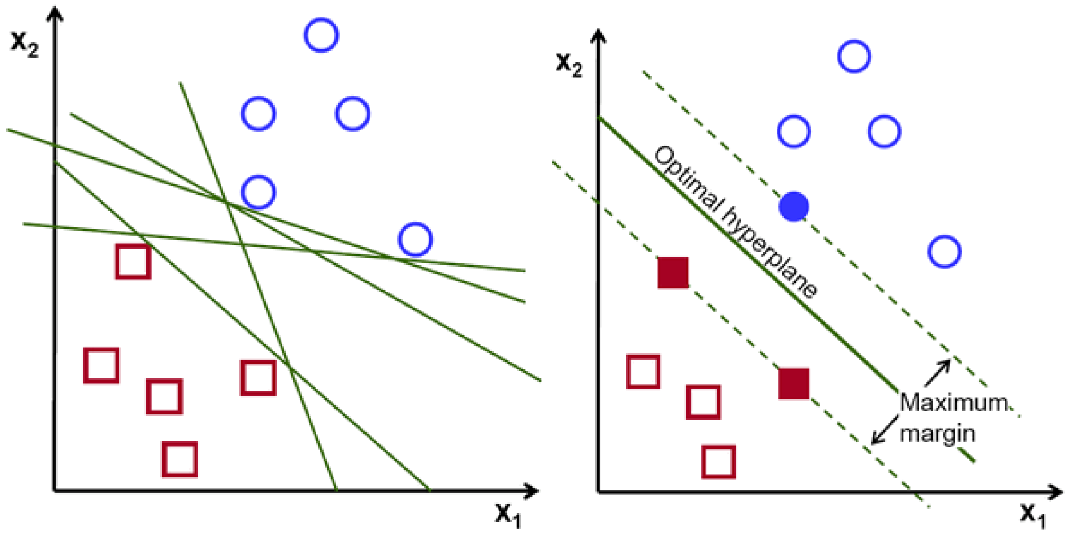
Training size/ Accuracy of Naive Bayes - Digit	10% -500	20% -1000	30% -1500	40% -2000	50% -2500	60% -3000	70% -3500	80% -4000	90% -4500	100% -5000
Iteration 1	66.2%	69.5%	69.0%	70.5%	70.5%	70.2%	69.4%	70.7%	70.7%	71.0%
Iteration 2	67.9%	69.4%	68.8%	69.7%	70.5%	70.9%	71.5%	70.5%	70.5%	71.0%
Iteration 3	67.3%	68.5%	69.6%	68.4%	69.9%	71.4%	71.1%	70.6%	70.4%	71.0%
Iteration 4	64.4%	67.0%	68.3%	71.1%	71.4%	69.9%	70.3%	70.5%	70.5%	71.0%
Iteration 5	65.0%	70.5%	70.5%	71.1%	70.5%	69.1%	70.9%	70.4%	70.5%	71.0%
Mean	66.5%	69.0%	69.2%	70.2%	70.6%	70.3%	70.6%	70.5%	70.5%	71.0%
Standard Deviation	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00





Support Vector Machines

In Support Vector Machines we aim to find a hyperplane in a N-D space (where n is the number of features) that distinctly classify the data point.



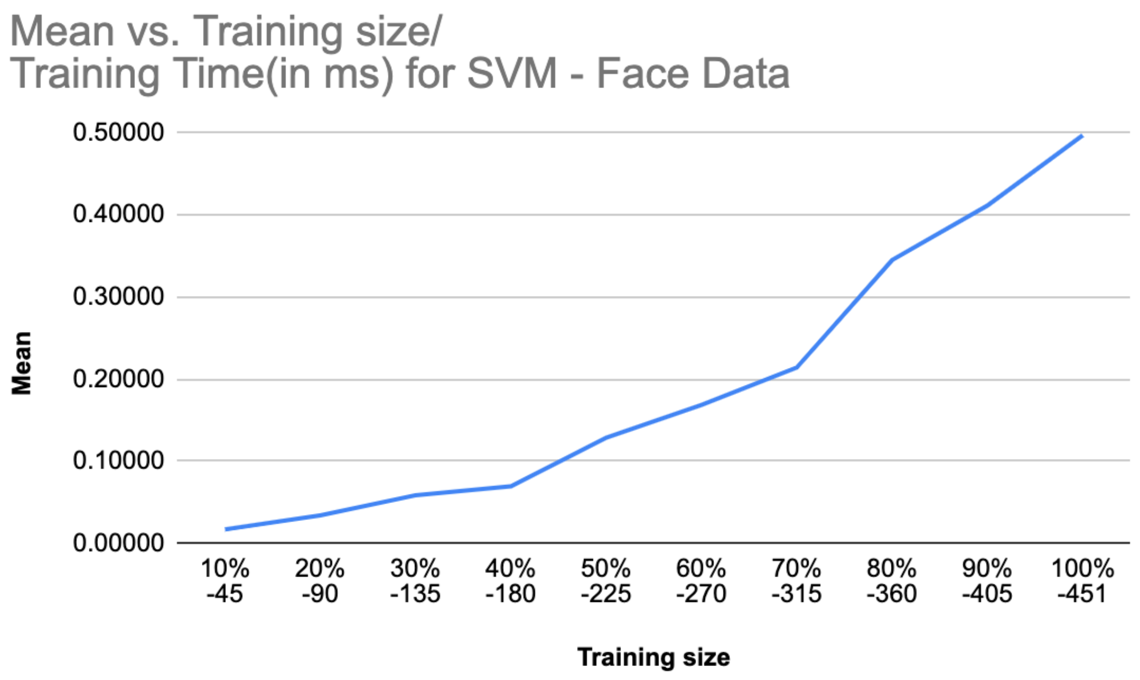
A variety of possible hyperplanes can be chosen to separate the two classes of data points. We aim to determine a place that has maximum margin so that the distance between data points of both the classes is the greatest. This way we get some reinforcement so that future data points can be classified with more reliance.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on both the sides of the hyperplane can be allocatedto different classes.
Support vectors are data points that are closer to the hyperplane. These data points can influence the position and orientation of the hyperplane.We maximize the margin of the classifier using these support vectors. The position of our hyperplane can change if we delete these support vectors.

Face Data:

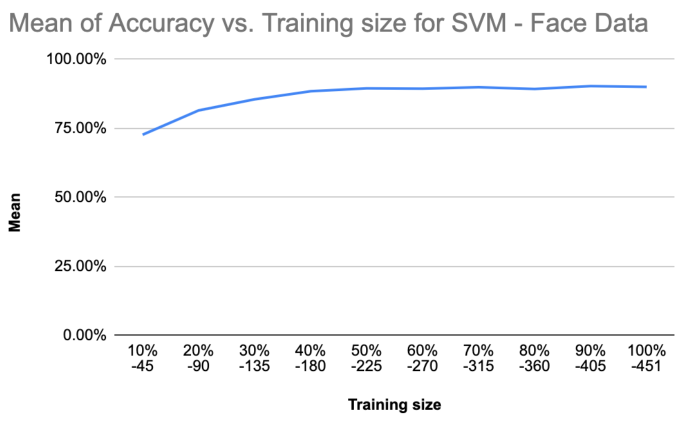
Training Time

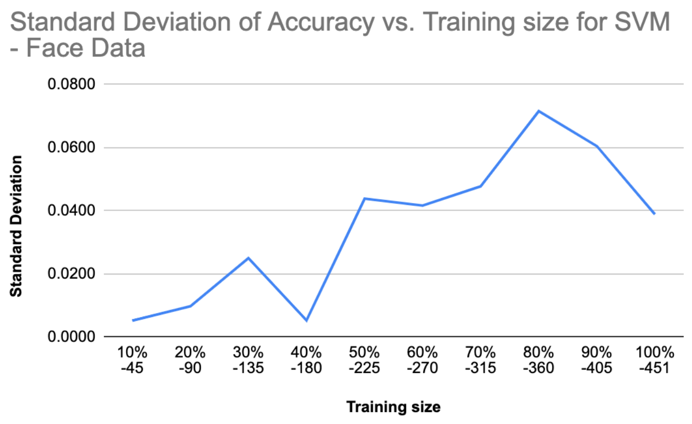
Training size/ Training Time(in s) for SVM - Face Data	10% -45	20% -90	30% -135	40% -180	50% -225	60% -270	70% -315	80% -360	90% -405	100% -451
Iteration 1	0.0257	0.0294	0.1031	0.0694	0.1224	0.2099	0.1616	0.2309	0.4620	0.4722
Iteration 2	0.0155	0.0337	0.0456	0.0667	0.1084	0.2073	0.2509	0.4248	0.3931	0.4711
Iteration 3	0.0172	0.0284	0.0459	0.0784	0.2053	0.1319	0.2745	0.3381	0.3511	0.5614
Iteration 4	0.0134	0.0276	0.0499	0.0657	0.0966	0.1733	0.1786	0.3752	0.3636	0.5069
Iteration 5	0.0132	0.0508	0.0487	0.0667	0.1108	0.1201	0.2050	0.3559	0.4875	0.4746
Mean	0.0170	0.0340	0.0587	0.0694	0.1287	0.1685	0.2141	0.3450	0.4115	0.4973
Standard Deviation	0.0051	0.0097	0.0249	0.0052	0.0438	0.0416	0.0477	0.0715	0.0604	0.0389



Accuracy

Training size/ Accuracy of SVM - Face	10% -45	20% -90	30% -135	40% -180	50% -225	60% -270	70% -315	80% -360	90% -405	100% -451
Iteration 1	78.7%	77.3%	84.0%	88.7%	89.3%	89.3%	90.7%	90.0%	90.7%	90.0%
Iteration 2	80.0%	78.0%	84.0%	88.7%	90.7%	88.0%	89.3%	89.3%	90.7%	90.0%
Iteration 3	65.3%	87.3%	85.3%	89.3%	89.3%	88.0%	88.7%	89.3%	89.3%	90.0%
Iteration 4	70.7%	84.0%	88.7%	85.3%	90.0%	91.3%	92.0%	89.3%	90.0%	90.0%
Iteration 5	68.7%	80.7%	85.3%	90.0%	88.0%	90.0%	88.7%	88.0%	90.7%	90.0%
Mean	72.7%	81.5%	85.5%	88.4%	89.5%	89.3%	89.9%	89.2%	90.3%	90.0%
Standard Deviation	0.0639	0.0420	0.0191	0.0180	0.0099	0.0141	0.0145	0.0073	0.0060	0.0000

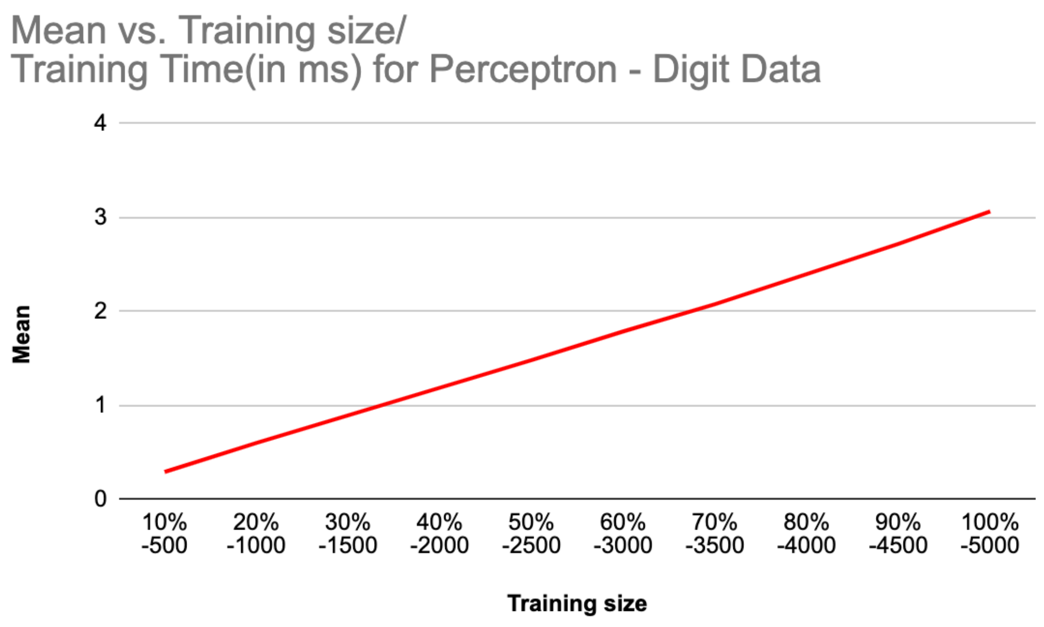




Digit Data:

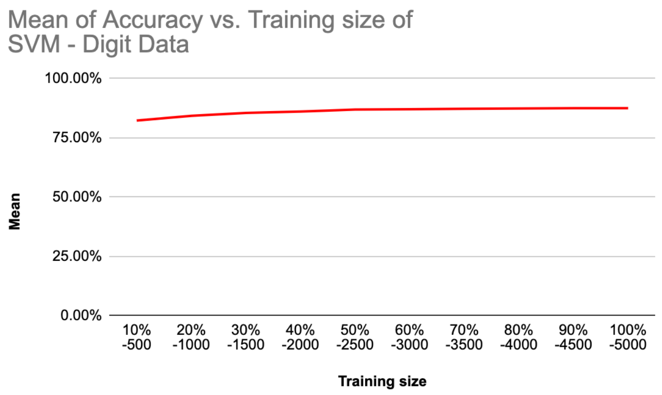
Training Time

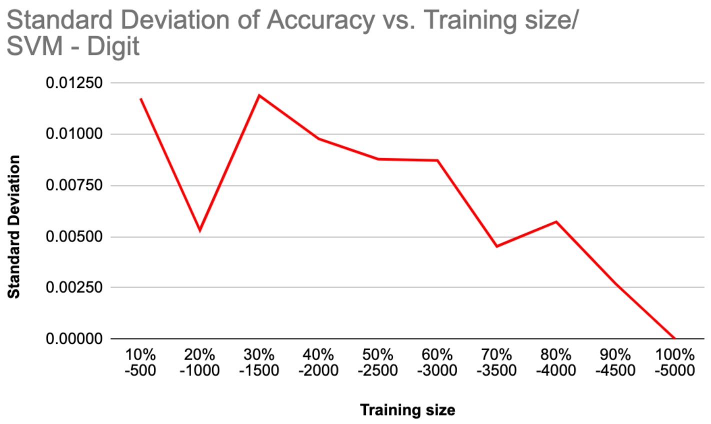
Training size/ Training Time(in s) for Perceptron - Digit Daa	10% -500	20% -1000	30% -1500	40% -2000	50% -2500	60% -3000	70% -3500	80% -4000	90% -4500	100% -5000
Iteration 1	0.2923	0.5997	0.8922	1.2101	1.4971	1.7969	2.0773	2.3904	2.7225	3.0172
Iteration 2	0.2913	0.5899	0.8876	1.1914	1.4628	1.7737	2.0843	2.3954	2.7003	3.1032
Iteration 3	0.2895	0.6036	0.9003	1.1665	1.4872	1.7769	2.0645	2.4000	2.7216	3.1495
Iteration 4	0.2903	0.5923	0.8862	1.2026	1.4669	1.7897	2.0752	2.4001	2.6974	3.0234
Iteration 5	0.2897	0.6044	0.9003	1.1668	1.4915	1.7952	2.0829	2.3976	2.7494	3.0225
Mean	0.2906	0.5980	0.8933	1.1875	1.4811	1.7864	2.0768	2.3967	2.7183	3.0632
Standard Deviation	0.0012	0.0066	0.0067	0.0201	0.0153	0.0106	0.0078	0.0040	0.0210	0.0600



Accuracy

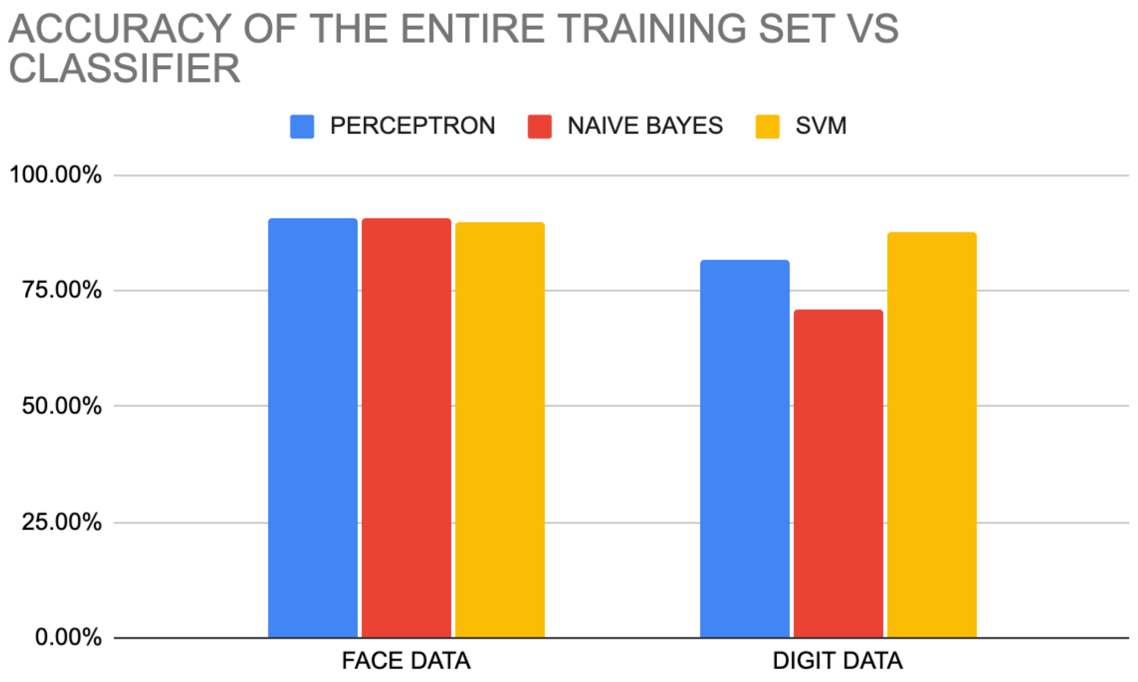
Training size/ Accuracy of SVM - Digit	10% -500	20% -1000	30% -1500	40% -2000	50% -2500	60% -3000	70% -3500	80% -4000	90% -4500	100% -5000
Iteration 1	81.6%	84.5%	86.1%	86.0%	86.7%	87.8%	86.8%	87.3%	87.8%	87.5%
Iteration 2	81.3%	84.3%	84.1%	85.0%	86.5%	86.8%	87.6%	86.7%	87.1%	87.5%
Iteration 3	83.8%	83.8%	87.0%	87.5%	86.5%	85.6%	87.2%	88.1%	87.5%	87.5%
Iteration 4	81.3%	83.7%	84.5%	86.5%	86.2%	87.4%	87.7%	87.7%	87.6%	87.5%
Iteration 5	83.2%	85.0%	85.8%	85.4%	88.4%	87.5%	86.7%	86.9%	87.3%	87.5%
Mean	82.2%	84.3%	85.5%	86.1%	86.9%	87.0%	87.2%	87.3%	87.5%	87.5%
Standard Deviation	0.0118	0.0053	0.0119	0.0098	0.0088	0.0087	0.0045	0.0057	0.0027	0.0000



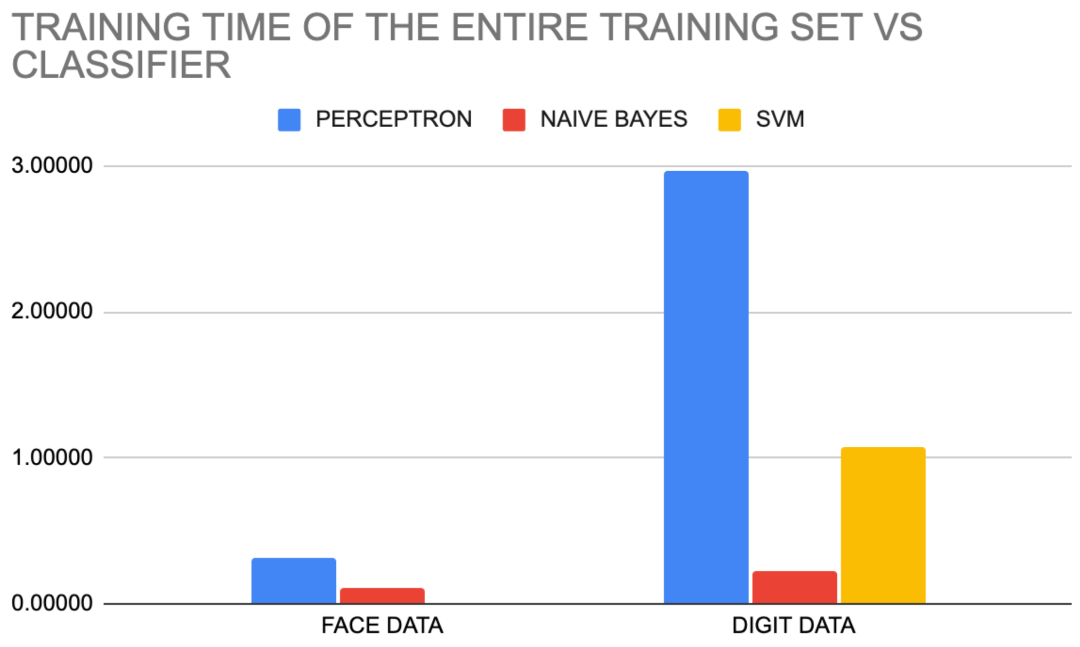


Results and Conclusions

- The accuracy on the Naive Bayes algorithm is approximately 71% for digits and 90.7% for faces. The perceptron is able to achieve 82.4% accuracy for digits, and 90.7% for faces. SVM performs the best out of the three algorithms for classifying digits with 87.5% accuracy, and is able to achieve 90.3% accuracy on faces.
- The accuracies improve linearly on sequentially increasing the training dataset.
- We observe here that both Perceptron and Naive Bayes perform better in the case of the face data whereas SVM gives a better accuracy than Perceptron and Naive Bayes in case of digit data.
- The difference in the accuracies between the face and the digit dataset is starkly different owing to the different dataset and label set sizes. Overall, faces data seems to achieve less prediction error most likely since it was a binary classification problem.



- The time taken to train the algorithm increases in a linear fashion on increasing the size of the training set.
- The time taken to train face data is immensely large as compared to the digit data simply because the size of the digit training data is significantly more as compared to face training data.
- It can also be noted that Perceptron takes comparatively more time to train than the other classifiers.


TRAINING TIME OF THE ENTIRE TRAINING SET VS CLASSIFIER
PERCEPTRON
NAIVE BAYES
SVM
FACE DATA
DIGIT DATA