AI5213/EC4213/CT5303/AI4021: Machine learning and deep learning **Spring 2023**

Report for Coding Assignment #1

(due on Wed May. 10, 2023. 11 :59PM)

Instructor: Jeany Son
Student name (GIST ID# / GitHub ID#): Jeong, Munki (20195163 / Munki-Jeong)

*REPORT1

1-1 : Report prior distributions for each class.

Since prior $P(Y = y) = \frac{the\ number\ of\ docs\ which\ label\ is\ y}{the\ number\ of\ entir\ docs}$, we have :

$$P(Y = ham) = \frac{the\ number\ of\ ham\ docs}{the\ numbe\ of\ entir\ docs} = \frac{3}{6} = \frac{1}{2}$$
$$P(Y = spam) = \frac{the\ number\ of\ spam\ docs}{the\ number\ of\ entire\ docs} = \frac{3}{6} = \frac{1}{2}$$

1-2 : Report likelihood distributions for each class.

Likelihood for each word w_i is,

$$P(w_i|Y = ham) = \frac{the\ total\ occurences\ of\ w_i\ in\ ham\ mails}{the\ totla\ occurences\ of\ all\ words\ in\ ham\ mails}$$

$$P(w_i|Y = spam) = \frac{the\ total\ occurences\ of\ w_i\ in\ spam\ mails}{the\ total\ occurences\ of\ all\ words\ in\ spam\ mails}$$

In Ham emails, the observed frequency of a given word w_i is incremented by a constant alpha (Here, alpha equals to 1) to apply laplace smoothing. Even if a word w_i appears in the Test email but not in any Ham email, its occurrence is still assumed to be as if it had been observed in a Ham email so that the count equlas to 1. By doing so, the Occurrences for each word can be calculated, and the likelihood for each word can be obtained. The result of this process concides with the result derived by Laplace smoothing formula with k=1.

$$P_{LAP,k}(x|y) = \frac{count(x, y) + k}{count(y) + k|X|}$$

The same thing apply to Spam mails so that obtain likelihood with smoothing.

Table 1 – Occurrences and Likelihood for Ham

Word	**Occurrences**	**Likelihood**	**Likelihood after Laplace Smoothing**
the	13	0.0751	0.0481
to	8	0.0462	0.0309
in	4	0.0231	0.0172
class	4	0.0231	0.0172
of	4	0.0231	0.0172
l best	4	0.0231	0.0172
dear	3	0.0173	0.0137
professor	3	0.0173	0.0137
am	3	0.0173	0.0137
your	3	0.0173	0.0137
...			
sum	172		

Table 2 – Occurrences and Likelihood for Spam

Word	**Occurrences**	**Likelihood**	**Likelihood after Laplace Smoothing**
you	12	0.0622	0.0376
to	8	0.0415	0.0260
will	5	0.0259	0.0173
the	5	0.0259	0.0173
for	4	0.0207	0.0145
can	4	0.0207	0.0145
your	4	0.0207	0.0145
of	4	0.0207	0.0145
it	4	0.0207	0.0145
we	3	0.0155	0.0116
...			
sum	193		

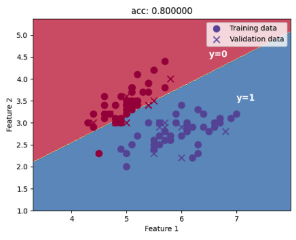
In report2, 3, and 7, the ratio (training data) :(validation data) = 4 :1 was used. For the rest, 7 :3 was used. Report 2, 3, 7 are problems with vanilla linear regression, and the rest deal with other models, so I believe that there is no logical problem with this manner.

REPORT2

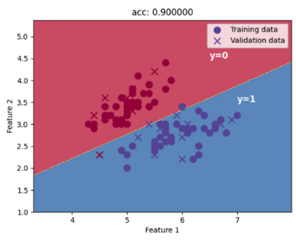
2-1 : Report the error and draw the decision boundary

I will present the results of five experiments each for maxiter=15 and maxiter=100. It was thought that there would be little room for meaningful analysis when the maxiter value is too small, as it rarely converges, or too large, as in almost all cases the accuracy becomes 1.0. However, I believed that the behavior of the results would vary depending on whether the maxiter value is small or large, and I chose 15 and 100 iterations as representative values that are "neither too small nor too large."

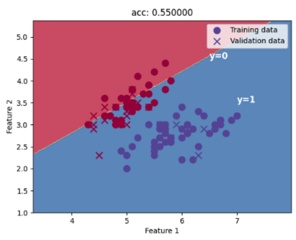
In these experiments, the eta value was set to 0.001. The reason for this choice is that subsequent experiments revealed that 0.001 and 0.01 are the optimal eta values. To minimize the influence of unoptimized eta values in this study, the optimal eta value of 0.001 was utilized.



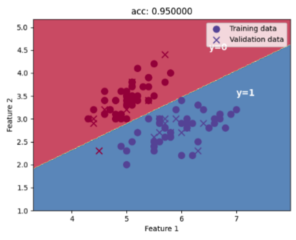
(a) 1st trial : acc=0.80



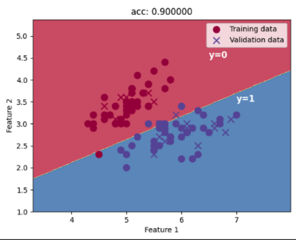
(b) 2nd trial : acc=0.90



(c) 3rd trial : acc=0.55

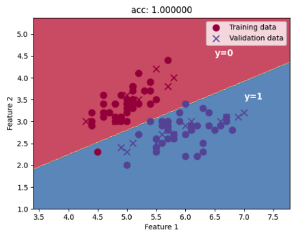


(d) 4th trial : acc=0.95

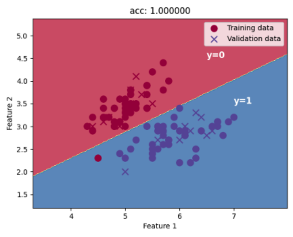


(e) 5th trial : acc=0.80

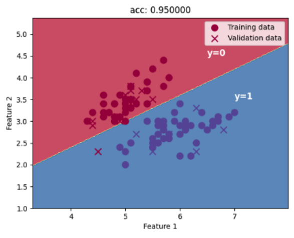
FIGURE 1 – Stocahstic ascent, maxiter=15



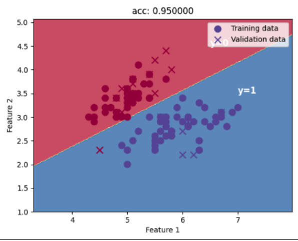
(a) 1st trial : acc=1.00



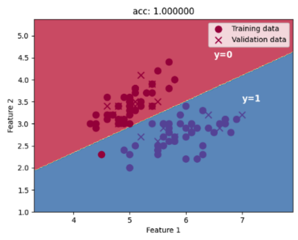
(b) 2nd trial : acc=1.00



(c) 3rd trial : acc=0.95



(d) 4th trial : acc=0.95



(e) 5th trial : acc=1.00

FIGURE 2 – Stocahstic ascent, maxiter=100

Consistent with what has been previously known theoretically, it can be observed that the accuracy improves as the maxiter value increases. One can also assert this in respect other than higher average accuracy. When the maxiter value is small, we can observe instances of outliers with extremely poor performance, as in the case of the 3rd trial with maxiter=15. On the other hand, such outliers were hardly observed when the maxiter value was sufficiently large. This tendencu (1 : better average accuracy as maxiter increases, 2 : fewer outliers) was confirmed not only in the data attached to this project but also in experiments with various cases such as maxiter=20, maxiter=30, and maxiter=110.

2-2 : Discuss the effect of the eta

In this section, the objective is to examine the impact of eta on accuracy, which involves investigating how accuracy changes as the eta value varies. Eta will be modified as follows : [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1].

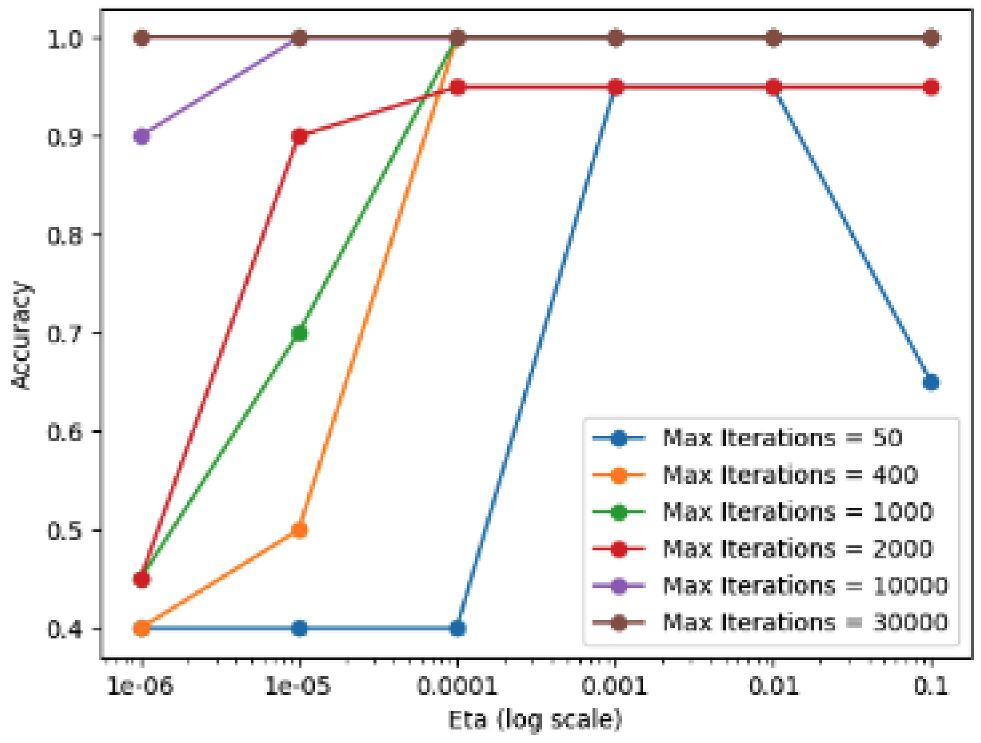


FIGURE 3 – Effect of Eta on accuracy for each MaxIter

Considering the accuracy behavior in relation to eta changes may differ depending on the given maxiter value, experiments were conducted for various maxiter values. The assigned maxiter values are 50, 400, 1000, 2000, 10000, and 30000.

The learning rate eta determines the size of the step the algorithm takes in each iteration when updating the weights. Smaller values cause the model to learn more slowly but can lead to more precise results, while larger values can make the model converge faster but risk surpassing optimal weights.

For each maxiter value, accuracy is plotted against different eta values on a logarithmic scale.

From the generated plot, the following observations can be made :

1. maxiter = 50

Very small eta values (0.000001, 0.00001, 0.0001) do not provide the model with sufficient time to converge, resulting in reduced accuracy. However, as eta increases (0.001, 0.01), the model can quickly converge and achieve high accuracy. When eta is too large (0.1), the model's accuracy declines. At first glance, it may appear to be a result of overshooting. Nevertheless, as this decrease is not evident for other maxiter values, it cannot be confidently attributed to overshooting.

2. maxiter = 400, 1000, 2000

The accuracy increases as eta changes from 0.000001, 0.00001, to 0.0001. This can be interpreted as sufficient iterations for updating weights leading to improved weights. The model can converge to high accuracy (0.95 for maxiter=2000, 1.00 for maxiter = 1000, 2000) starting from eta values of 0.0001.

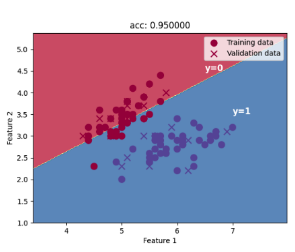
3. maxiter = 10000, 30000

For very large maxiter values (e.g., 10000, 30000), the model consistently achieves high accuracy across all eta values. This indicates that given enough iterations, the model can find the optimal weights irrespective of the selected learning rate. None of the listed eta values are excessively large to provoke overshooting, in my opinion.

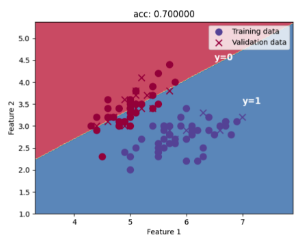
In summary, for logistic regression using gradient ascent, an appropriate eta value can help the model achieve high accuracy in fewer iterations. However, in this experiment, if there are enough iterations, the model can ultimately converge to high accuracy even for best eta values. Eta values of 0.001 and 0.01 demonstrated good performance regardless of maxiter value, and I believe these values were trustworthy to be the optimal eta values for my stochastic gradient model.

REPORT3

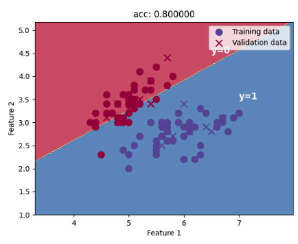
3-1 : Report the error and draw the decision boundary I test my model with eta= 0.001, maxIter = 80, the ratio of (training data) :(test data) = 8 :2 Some of trials show good results(such as 1st, 4th, 5th), as the results



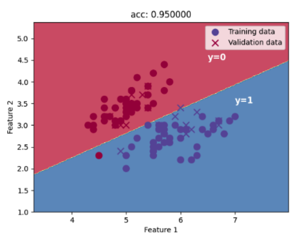
(a) 1st trial : acc=0.95



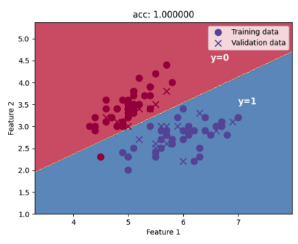
(b) 2nd trial : acc=0.70



(c) 3rd trial : acc=0.80



(d) 4th trial : acc=0.95

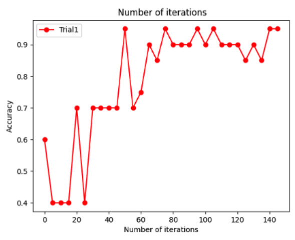


(e) 5th trial : acc=1.00

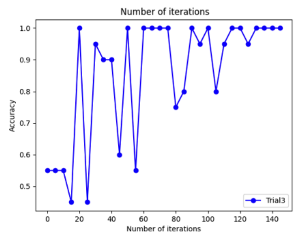
FIGURE 4 – Stochastic gradient ascent.

above. Compared to the gradient model, which showed consistent good results with the values about maxIter = 60, the SGA model falls short of that. If the batchsize(which is randomly selected during each iteration)is small, the contribution to the weight update will be small. When the maxiter is small, each case with the small batchsize will be greatly influenced to overall accuracy.

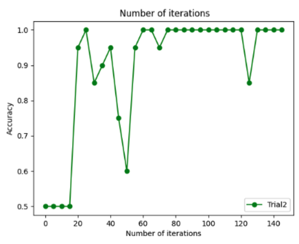
3-2 : The effect of maxIter on the accuracy, given eta=0.001 In Report 2, one of the identified optimal eta values, eta = 0.01, was used to conduct the experiment. The reason for using the optimal eta is to observe the behavior of accuracy with respect to the changes in maxIter while minimizing the influence of external factors.
The change in accuracy was observed by incrementing maxIter from 5 to 145, in intervals of 5. This trial was repeated three times, and the results are as follows. The results of these experiments can be analyzed in three phases :



(a) 1st trial



(b) 2nd trial



(c) 3rd trial

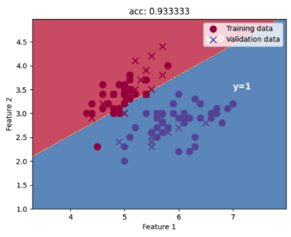
FIGURE 5 – Stochastic gradient ascent.

1 : Very small maxIter, such as $maxIter < 20$: In all cases, extremely low accuracy is observed without exception.
2 : Intermediate values, such as $20 < maxIter < 80$: While some instances show good accuracy, consistency is lacking.
3 : Sufficiently large values, such as $80 < maxIter$: Except for the occasional outliers, consistently good accuracy is observed within a certain range.

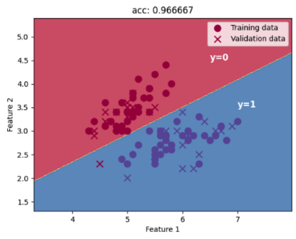
REPORT4

4-1 : Report the error and draw the decision boundary

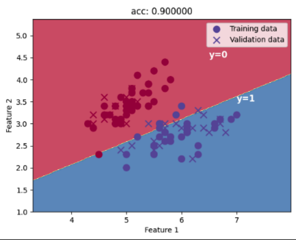
I test my model with eta= 0.001, maxIter = 100, lambda = 0.1, the ratio of (training data) :(test data) = 7 :3 The reason why I select lambda = 0.1 is, 0.1 is one of values that show stable result so that tendency of the model(According to 4-2). It shows quite good accuracy, as the results above.



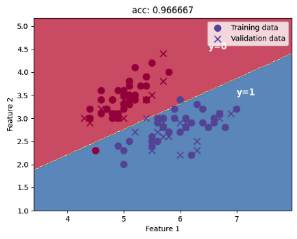
(a) 1st trial : acc=0.93



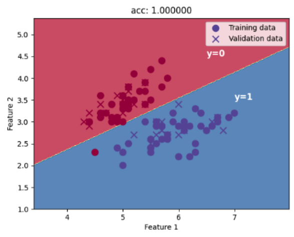
(b) 2nd trial : acc=0.96



(c) 3rd trial : acc=0.90



(d) 4th trial : acc=0.96

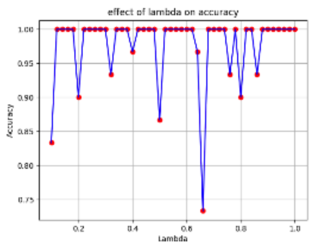


(e) 5th trial : acc=1.00

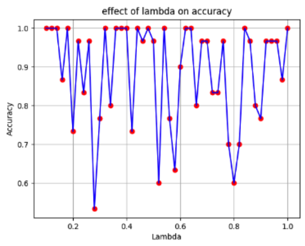
FIGURE 6 – Regularized logistic regresson.

4-2 : The effect of lambda on accuracy

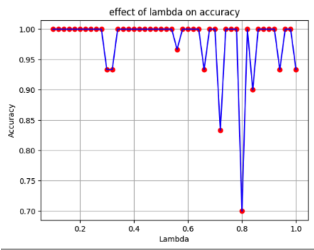
I examined the results at intervals of 0.02 from 0.1 to 1.0, and at intervals of 1.00 from 1.0 to 5.0. To control for factors other than lambda, the same tr-x, tr-y, val-x, and val-y were used within each trial. Additionally, in Figure 5-a and 5-d, the same data was used to investigate the results in different ranges. In most trials (including those not



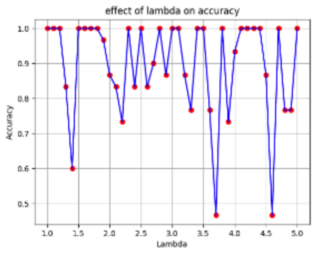
(a) 1st trial : range from 0.1 to 1.0



(b) 2nd trial : range from 0.1 to 1.0



(c) 3rd trial : range from 0.1 to 1.0



(d) range from 1.0 to 5.0

FIGURE 7 – The effect of lambda on accuracy

presented here), 2) for lambda values up to 0.6, respectable accuracy is displayed for almost all lambda values. Of course we can find some cases where the results are generally unsatisfactory, such as in Figure 5-(b), where the results are generally unsatisfactory ; however, even in these cases, relatively decent results can be observed for values less than 0.6.
In contrast, there are hardly any trials that exhibit stable results for lambda values greater than 0.6. Although some stable regions appear in Figure 5-(a), trials exhibiting such cases are quite rare.

4-3 : Compare two cases : when lambda=0, lambda = 0.25, lambda = 0.5

To examine the effect of lambda on regularization, I aimed to compare the results when lambda is 0, 0.25, and 0.5, using the same settings in each trial. Since it was confirmed in Section 4-2 that stable results are obtained for lambda values up to 0.6, it was deemed acceptable to use 0.25 and 0.5 as values for comparison.
A total of 100 trials were conducted, and the average and variance results for each lambda are presented in the table below. Unfortunately, I failed to discover any significant results in terms of regularization as anticipated. The reason

Table 3 – Average and Variance of Accuracy for Different Lambda Values of 100 number of trials

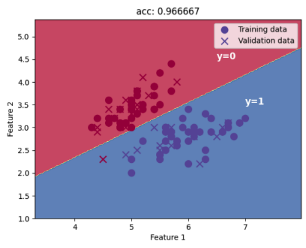
	Average	Variance
Lambda = 0	0.96667	0.00094
Lambda = 0.25	0.95000	0.00125
Lambda = 0.5	0.96667	0.00069

for this is likely due to 1) the dataset size not being large enough to induce overfitting.

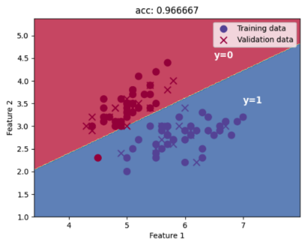
REPORT5

5-1 : Report the error and draw the decision boundary

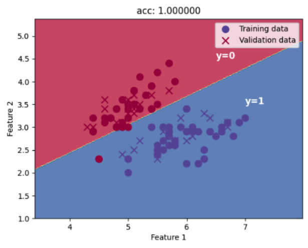
I test my model with maxEpoch = 60, threshold = 2, the ratio of (training data) :(test data) = 7 :3



(a) 1st trial



(b) 2nd trial

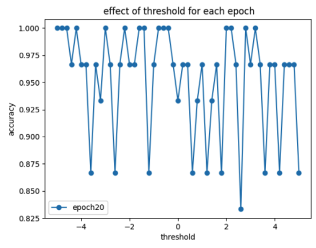


(c) 3rd trial

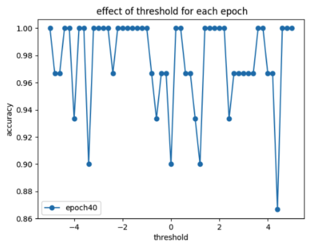
FIGURE 8 – Decision boundary

It shows quite good accuracy, as the results above.

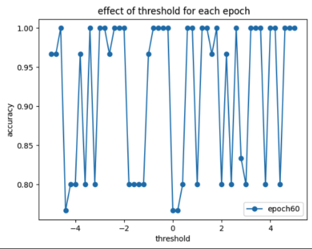
5-2 : The effect of threshold on the accuracy



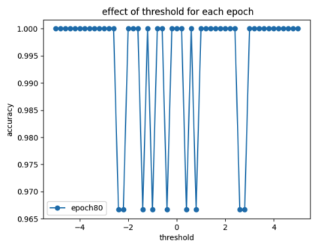
(a) maxepoch = 20



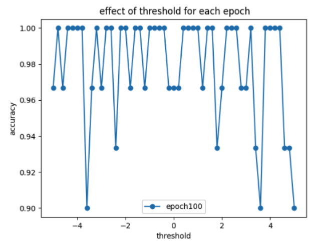
(b) maxepoch = 40



(c) maxepoch = 60



(d) maxepoch = 80



(e) maxepoch = 100

FIGURE 9 – effect of threshold on accuracy for each maxepoch

Observing the result above, there is no meaningful correlation between threshold value and accuracy. Especially, when maxepoch(the number of iteration)is sufficiently large(for maxepoch=60, 80 in this case), it shows good accuracy for most of the cases.
This conclusion applies to threshold=0. However, threshold=0 would have some special meaning theoretically(i.e find meaning from math.) Using zero-threshold means the decision depends solely on $\vec{W} \cdot \vec{X}$. That is, the decision is based on whether dot product is positive or negative.
This could affect performance because we cannot expect effect of bias term. That is, the model with zero-threshold does not account for potential imbalances or noise in the data. Nonetheless, this would be no really matter when maxepoch is sufficiently large so that a model learn proper weights. On the other hand, proper bias term balance noise in data when maxepoch is small(or training data is not sufficient)

Intro to Report 7, 8, 9

I compare the trainGA, trainSGA, train-reg-SGA and correspoding Scikit-learn's logistic regression models(respectively) from three perspectives. Basically, I compared my own model implentations and those in scikit-learn in terms of accuracy, as used in Report2-5. I employed the previously implemented computeClassifica-tion method for my models(trainGA, trainSGA), and the accuracy-score method. I used the results obtained when the iris data was split at a ratio of 4 :1 for report 7, and 7 :3 for report 6, 8, and 9(training dataset : validation dataset). However, I felt that fixing the split ratio by arbitrarily determining value, and proceeding in this manner might not lead to a complete comparison. This led for me to introduce the second and third perspectives, which are cross-validation and learning-curve methods.
Initially, I intended to utilize cross-val-score and learning-curve from sklearn.model-selection, but discovered that both of these methods can only be applied to the LogisticRegression.fit() model defined in sklearn. Consequently, I used customized functions that possessed the same functionality but could be applied to other models as well.
During the test, cross validation test will be proceeded based on cv=5. Also, learning curve test will be proceeded based on training sizes = [0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9], cv=5.

REPORT6

Both my model and scikit-learn model predict test mail as 'ham'. However, there is difference in posteriori porbability.

Table 4 – Posteriori prob

	My naive bayes model	Scikit-learn Model
Ham	2.932470596614825432924920475E-111	0.99997261
Spam	6.425266301703030089922473157E-119	2.7387182e-05

Though they predict same result, there is difference in numerical value.
There are huge differences in postprobability of ham and spam. This would be caused by scikit learn model not applying laplace smoothing. I think there would be some other factors. For example, different preprocessing steps would impact on the value.

REPORT7

First perspective : Accuracy

TABLE 5 – Model performance comparison

	Gradient ascent	Stochastic Gradient ascent	Scikit-learn Model
Accuracy	0.95	0.9	1.0

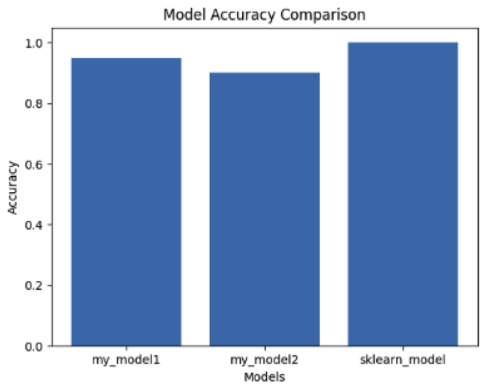


FIGURE 10 – accuracy

The scikit-learn model achieves perfect accuracy, while both custom models have slightly lower accuracy. This indicates that the scikit-learn model is better at the validation set.

Second perspective : cross-validation
Perform k-fold cross-validation, where the dataset is divided into equally sized cv sets (here, cv=5). The scikit-learn

TABLE 6 – Model performance comparison

	Gradient ascent	Stochastic Gradient ascent	Scikit-learn Model
Cross-validation scores	[1.0, 1.0, 0.95, 0.95, 1.0]	[1.0, 1.0, 0.95, 0.95, 1.0]	[1. 1. 1. 1. 1.]

model consistently achieves perfect accuracy across all 5 cross-validation folds, whereas the custom models have slightly lower scores in some folds. This again indicates that the scikit-learn model has better performance.

Third perspective : learning-curve
Vary the proportion of the total data used as training data, and perform the same procedure as in cross-validation for each ratio, using the corresponding training dataset (with cv=5, as before).

TABLE 7 – Learning curve validation scores comparison

Training Set Sizes	GA	SGA	Scikit-learn Model
0.6	[0.975, 1.0, 1.0, 0.975, 0.975]	[1.0, 0.975, 1.0, 1.0, 1.0]	[0.9, 0.8, 0.95, 0.85, 0.65]
0.65	[1.0, 0.9714, 0.9714, 1.0, 1.0]	[1.0, 1.0, 1.0, 1.0, 0.9714]	[0.9, 0.9, 0.95, 0.85, 0.9]
0.7	[0.9667, 0.9667, 1.0, 0.9667, 1.0]	[1.0, 0.9667, 1.0, 0.9667, 1.0]	[0.9, 0.95, 1.0, 0.9, 1.0]
0.75	[1.0, 1.0, 0.96, 1.0, 0.96]	[1.0, 1.0, 1.0, 1.0, 1.0]	[0.9, 1.0, 1.0, 0.95, 1.0]
0.8	[1.0, 1.0, 1.0, 1.0, 0.95]	[1.0, 1.0, 1.0, 1.0, 1.0]	[0.9, 1.0, 1.0, 0.95, 1.0]
0.85	[1.0, 1.0, 1.0, 1.0, 1.0]	[1.0, 0.933, 0.933, 1.0, 1.0]	[0.9, 1.0, 1.0, 0.95, 1.0]
0.90	[0.9, 0.9, 1.0, 1.0, 1.0]	[1.0, 1.0, 1.0, 1.0, 0.9]	[1.0, 1.0, 1.0, 0.95, 1.0]

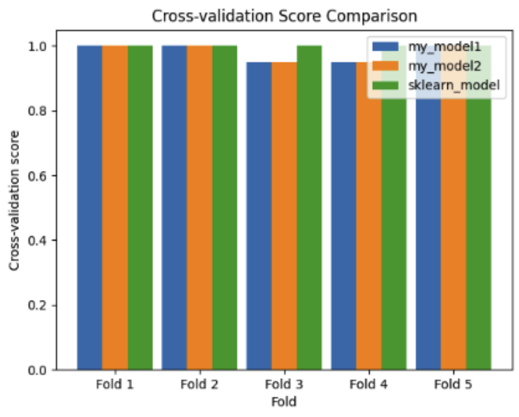


FIGURE 11 – cross-validation

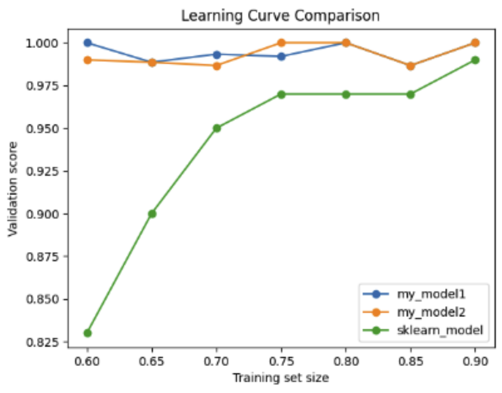


FIGURE 12 – learning-curve

When comparing the learning curves, both custom models perform better than the scikit-learn model especially for smaller training set sizes. As the training set size increases, the scikit-learn model's performance improves, and it becomes comparable to the my models. Stochastic Gradient Ascent appears to perform slightly better than Gradient Ascent for some training set sizes, and Gradient Ascent is better for the other set sizes.

Inferring the Reason for performance differences

In conclusion, the scikit-learn model outperforms the custom models in terms of accuracy and cross-validation scores. Possible reason for difference :
- Scikit-learn uses the LBFGS solver, which converges faster and provides better results than the Gradient Ascent (GA) and Stochastic Gradient Ascent (SGA).
- Automatic step-size adaptation : Unlike gradient ascent and stochastic gradient ascent, which requires a fixed or manually tuned step size (learning rate), LBFGS can automatically adapt the step size during optimization. This makes it easier to find a good learning rate and can further improve the convergence speed and the quality of the final solution.

However, in the learning curve approach, the custom models perform relatively well, particularly stochastic gradient ascent, and even outperform the scikit-learn model at smaller training set sizes.
The scikit-learn model basically possess penalty term in objective function. This means more robust and less prone to overfitting. It could be that the scikit-learn model is more unwilling to fit into smaller datasets so that prevent overfitting, which may lead to slightly worse performance at smaller training set sizes in the learning curve analysis.

REPORT8

First perspective : Accuracy

From Table above, the Scikit-learn model has a higher accuracy (1.0) than the custom model (0.83). This indicates

Table 8 – Accuracy comparison

	Custom Model	Scikit-learn Model
Accuracy	0.8333	1.0

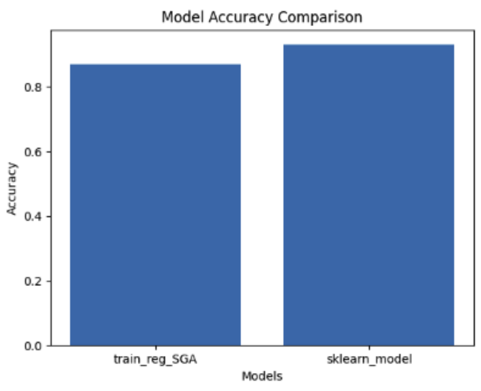


Figure 13 – learning-curve

that the Scikit-learn model performs better in terms of classification accuracy on the validation set.

Second perspective : cross-validation

Comparing the cross-validation scores in the table above, the sklearn model consistently achieves higher scores across

Table 9 – Cross-validation scores comparison

	Custom Model	Scikit-learn Model
Cross-validation scores	[1.0, 1.0, 0.95, 0.95, 1.0]	[1.0, 1.0, 1.0, 1.0, 1.0]

all folds, with an average score of 1.0. The custom model has a slightly lower average score, with one fold achieving 0.95. This suggests that the sklearn model is more robust and generalizes better to new data.

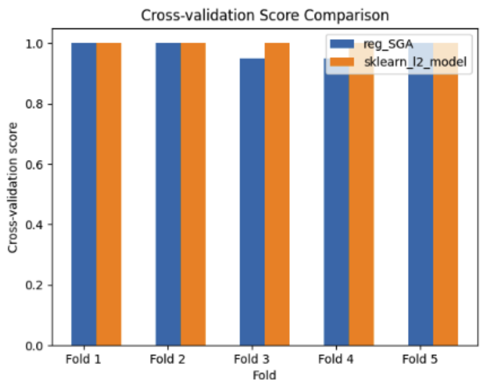


Figure 14 – cross validation

Third perspective : learning-curve

First of all, for all cases, the accuracy of the sklearn model is lower. Particularly, for cases with training data ratios

Table 10 – Learning curve validation scores comparison

Training Set Sizes	Custom Model	Scikit-learn Model
0.6	[0.975, 1.0, 0.975, 1.0, 0.975]	[0.9, 0.8, 0.95, 0.85, 0.65]
0.65	[1.0, 1.0, 0.9714, 1.0, 1.0]	[0.9, 0.9, 0.95, 0.85, 0.9]
0.7	[0.9667, 1.0, 1.0, 1.0, 0.9667]	[0.9, 0.95, 1.0, 0.9, 1.0]
0.75	[1.0, 1.0, 1.0, 0.96, 0.96]	[0.9, 1.0, 1.0, 0.95, 1.0]
0.8	[1.0, 1.0, 1.0, 1.0, 1.0]	[0.9, 1.0, 1.0, 0.95, 1.0]
0.85	[1.0, 0.9333, 1.0, 1.0, 0.9333]	[0.9, 1.0, 1.0, 0.95, 1.0]
0.9	[1.0, 1.0, 1.0, 1.0, 1.0]	[1.0, 1.0, 1.0, 0.95, 1.0]

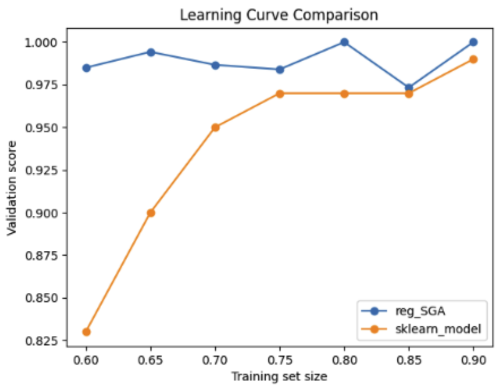


Figure 15 – learning-curve

of 0.6 and 0.65, we can observe relatively low accuracy.

Inferring the Reason for performance differences

In terms of accuracy and cross-validation, there is only a slight difference, which makes comparison less meaningful. On the other hand, the relatively poor performance of the sklearn model when the training data ratio is small in the learning curve may imply that it possesses a good regularizer. A good regularizer exists to avoid overfitting. The fact that the sklearn model does not fit well when the dataset size is very small, indicates that it has a good performance in avoiding overfitting, and can be interpreted as having a good regularizer.

REPORT9

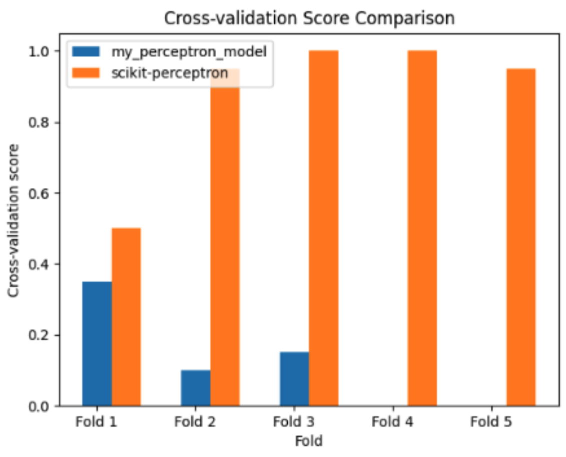
First perspective : Accuracy

This comparison is based on experiments with maxepoch = 100 and threshold = 2. Moreover, Training data :Validation data = 7 :3

Table 11 – Accuracy

My perceptron model	Scikit-learn Model
0.93	0.97
0.97	0.97
0.97	0.93
0.97	0.97
0.97	0.97

When considering only accuracy, there is no significant performace difference between this two model.



Second perspective : cross-validation

Table 12 – Cross-validation scores comparison

	Custom Model	Scikit-learn Model
Cross-validation scores	[0.35, 0.1, 0.15, 0.0, 0.0]	[0.5, 0.95, 1.0, 1.0, 0.95]

Examining the cross-validation results, my perceptron model records extremely low accuracy compared to the scikit-perceptron model. When using cross-validation, the dataset size is necessarily small because we 'fold' datasets for each trial. As mentioned in Report 5, a proper threshold, which serves as a bias term, can play a significant role when the data size is not sufficient.

In contrast to my model's fixed threshold acting as a bias, the scikit-learn model can flexibly determine the intercept term, which may explain its better performance with small dataset sizes. Furthermore, from this perspective, it is likely that the scikit-learn model is also beneficial for avoiding overfitting(one of the primary reasons for using cross-validation).

I will not proceed with Learning curve approach since the comparison on second perspective is sufficient to reveal the difference of this two models.