SMAI Assignment 2

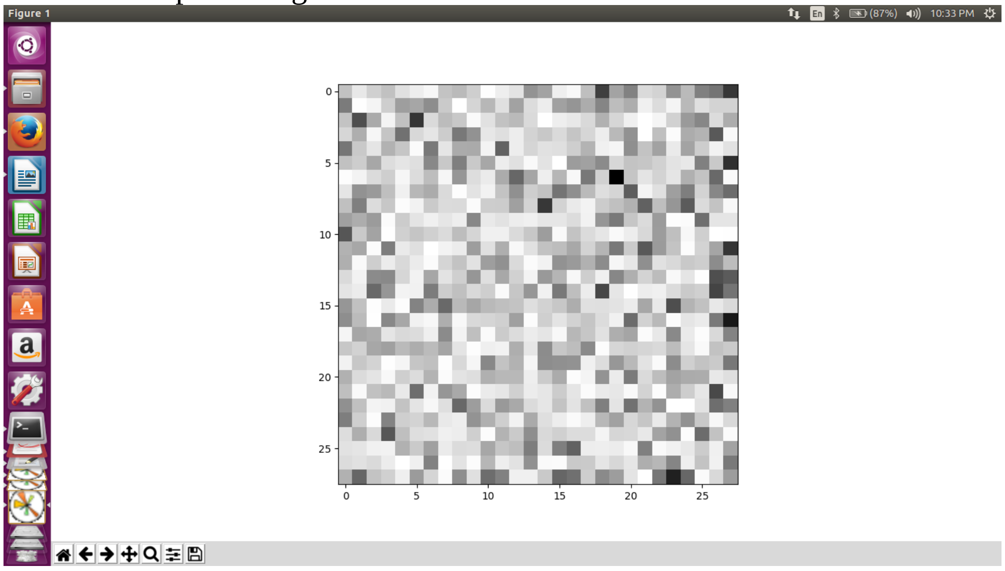
Naila Fatima
201530154

Problem 3: Logistic Regression and Regularization

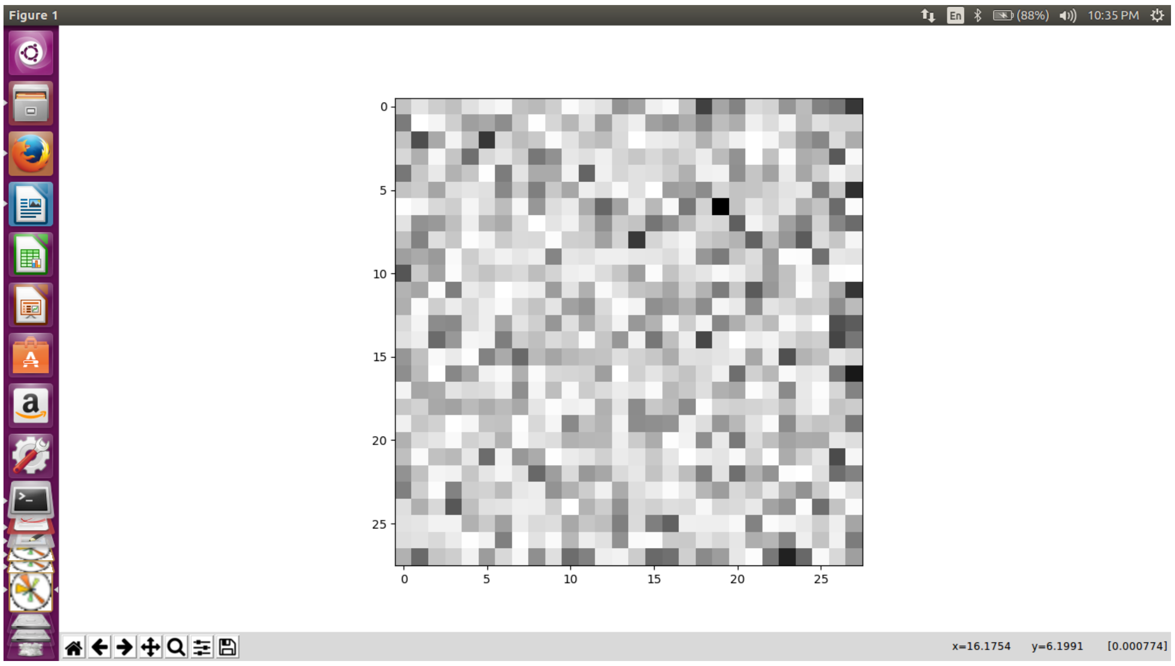
Note that the hyperparameter used by the sklearn model is C which is the inverse of lambda. We know that as the lambda value (smaller the C value) increases, the chance of the model overfitting the training data decreases. This means that the larger the value of lambda, the lesser is the chance that the model will change in order to accomodate outliers.

L2 Regularization:

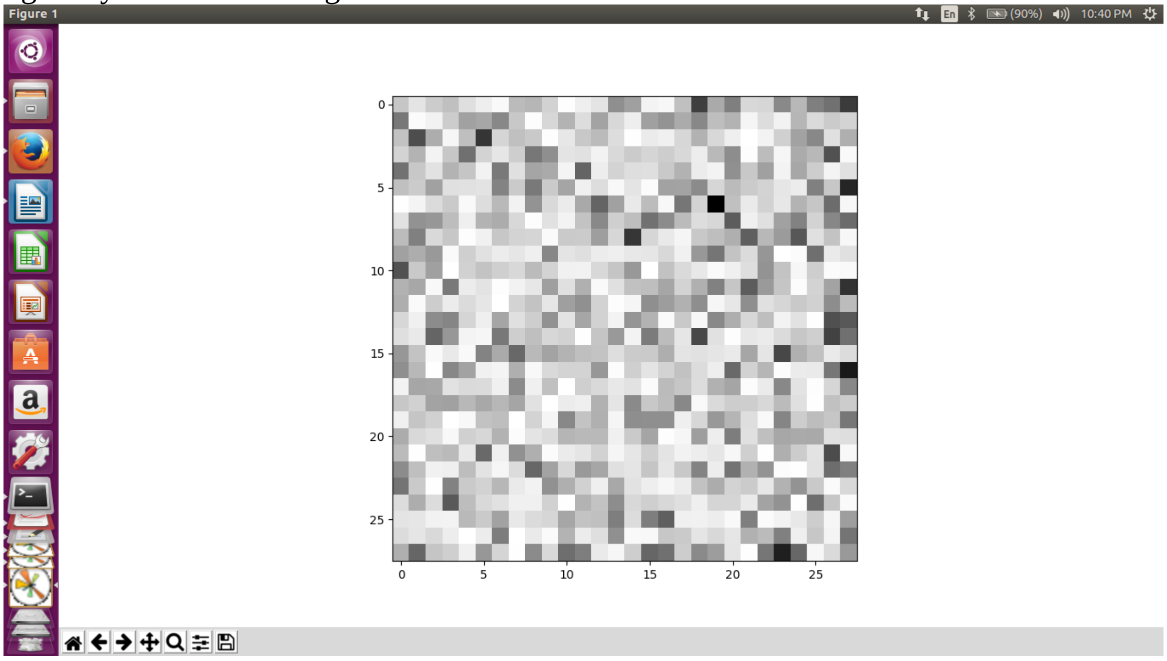
Using the standard logistic regression model of sklearn with L2 regularization and C = 1, the accuracy, precision, recall and f1 scores are 0.936832740214, 0.93848857644991213, 0.93684210526315792 and 0.93766461808604051, respectively. The weights are of the order of 10^{-3} and 10^{-4}. Activation map for weights is shown below.



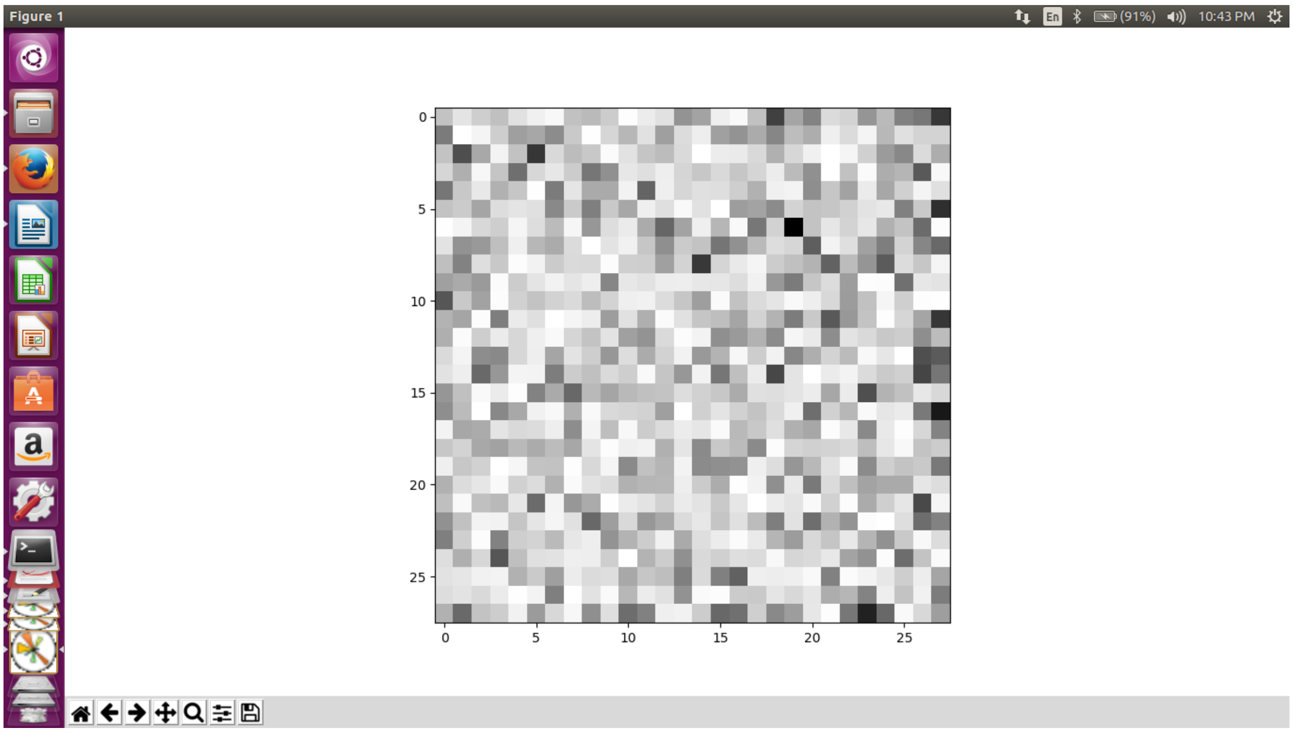
On decreasing C to 0.5 (so lambda will increase to 2), the accuracy, precision, recall and f1 scores become 0.935053380783, 0.93520140105078808, 0.93684210526315792 and 0.93602103418054339. We can see that all these, except recall which has not changed, have decreased on decreasing C. The weights have increased in *magnitude* on decreasing C to 0.5, although the exponent has remained same.



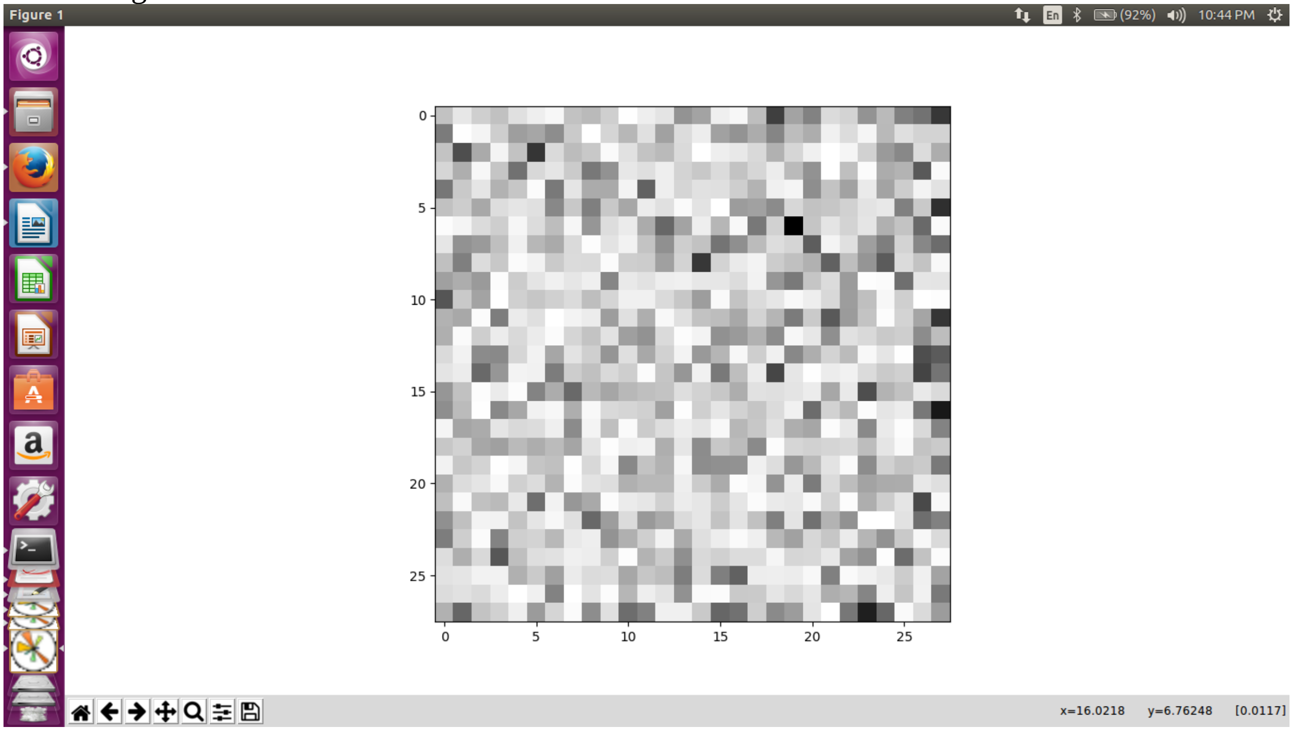
On decreasing C further to 0.001, the accuracy, precision, recall and f1 scores have become 0.936832740214, 0.93542757417102962, 0.94035087719298249 and 0.93788276465441822. We can see that on comparing this with the case when C was 1, the accuracy has remained the same and the precision has dropped marginally but both recall and f1 scores have increased slightly. The weights have decreased when compared to the previous case of C = 0.5 in *magnitude* although the exponent has remained same. The change in weights is very marginal and it is of the type where the magnitude of the weights has increased. For example, if in case of C = 0.5, the weights were -4.2 x 10^{-3} and 2.3 x 10^{-3}, in this model, the weights have decreased in magnitude to around -2.1 x 10^{-3} and 1.6 x 10^{-3}. Note that the weights for C = 1 are more closer to those of C = 0.001 as compared to 0.5, although they are still a bit larger.



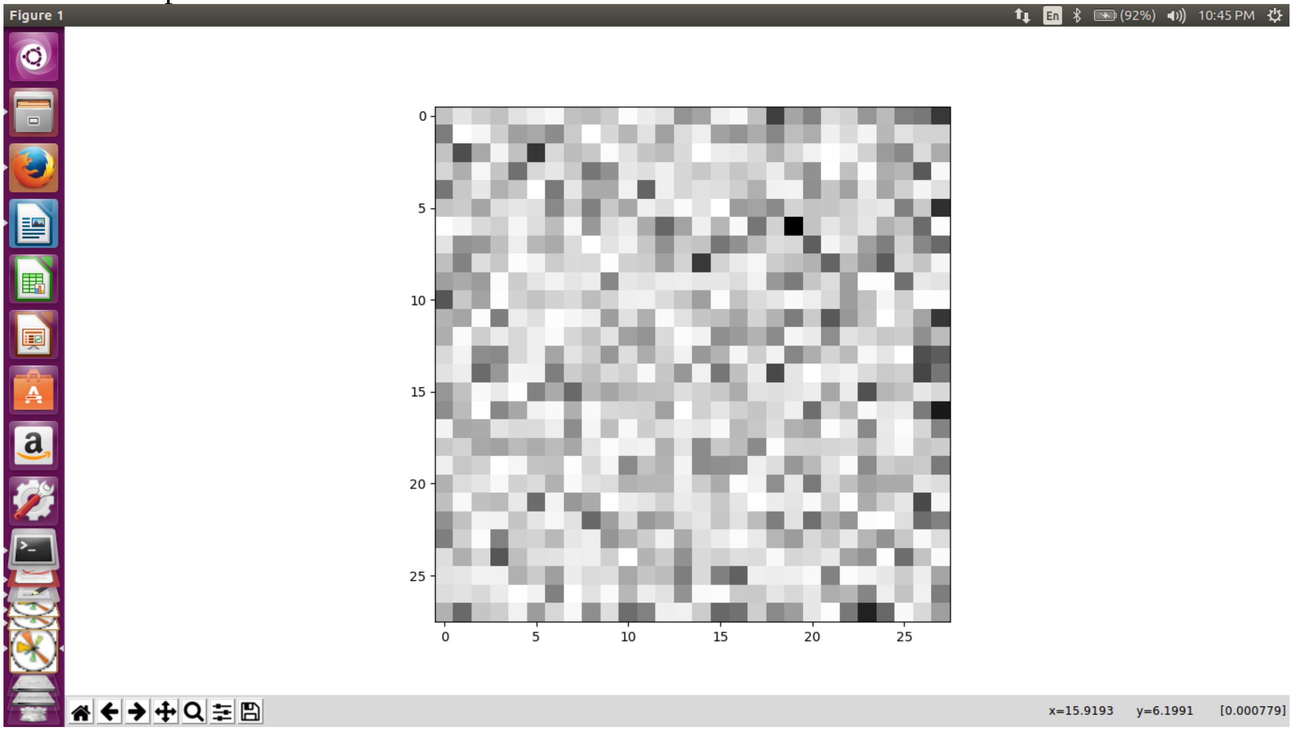
On increasing C to 1.25, the accuracy, precision, recall and f1 scores are 0.935943060498 ,0.93684210526315792, 0.93684210526315792 and 0.93684210526315792. We can see that all the metrics except the recall (which has remained the same) have decreased marginally compared to C = 1. Also, the weights have once again increased in *magnitude*.



On increasing C to 1.5 (decreasing lambda), the accuracy, precision, recall and f1 scores are 0.935053380783, 0.93520140105078808, 0.93684210526315792 and 0.93602103418054339. Some weights have increased while some others have decreased.

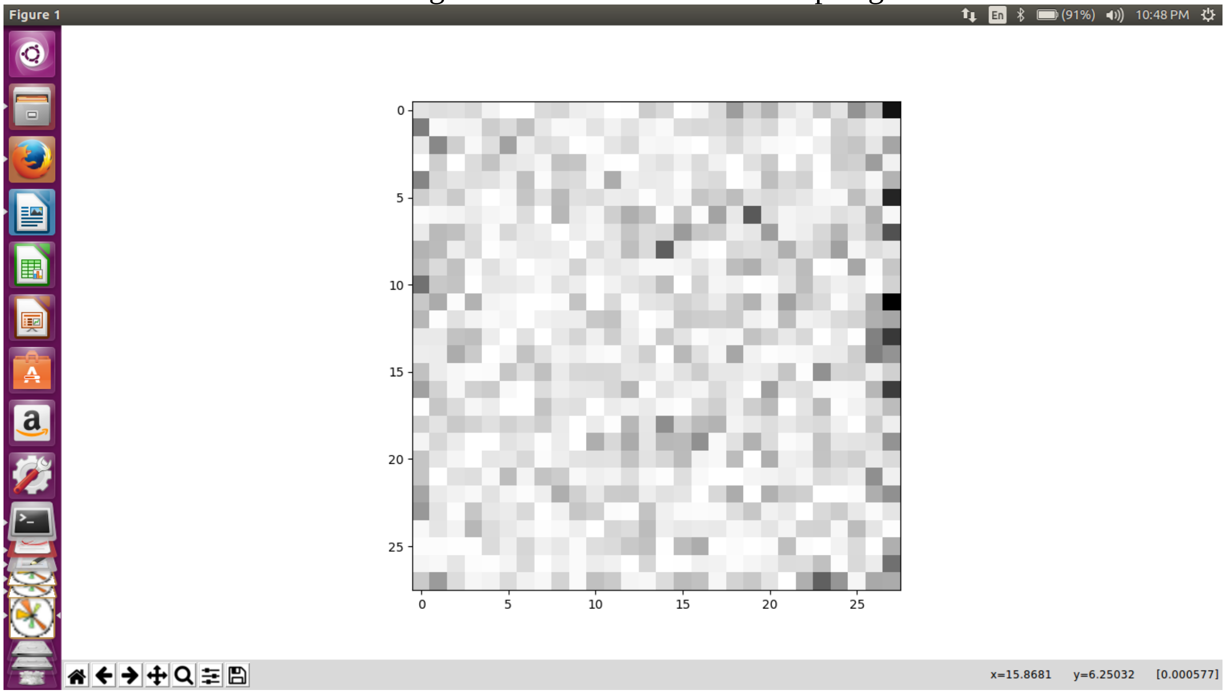


On increasing C to 10, the accuracy, precision, recall and f1 scores are now 0.934163701068 ,0.93356643356643354, 0.93684210526315792 and 0.93520140105078808 . We can see that all the metrics except the recall have decreased.

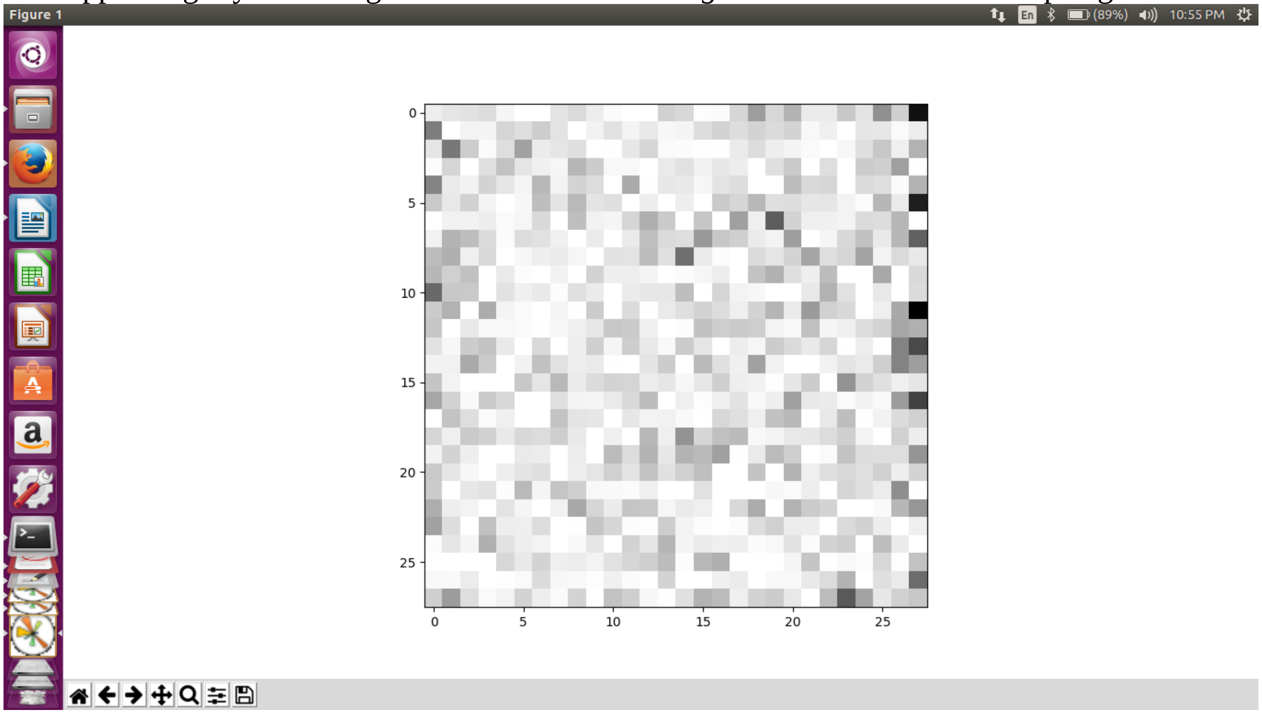


L1 Regularization:

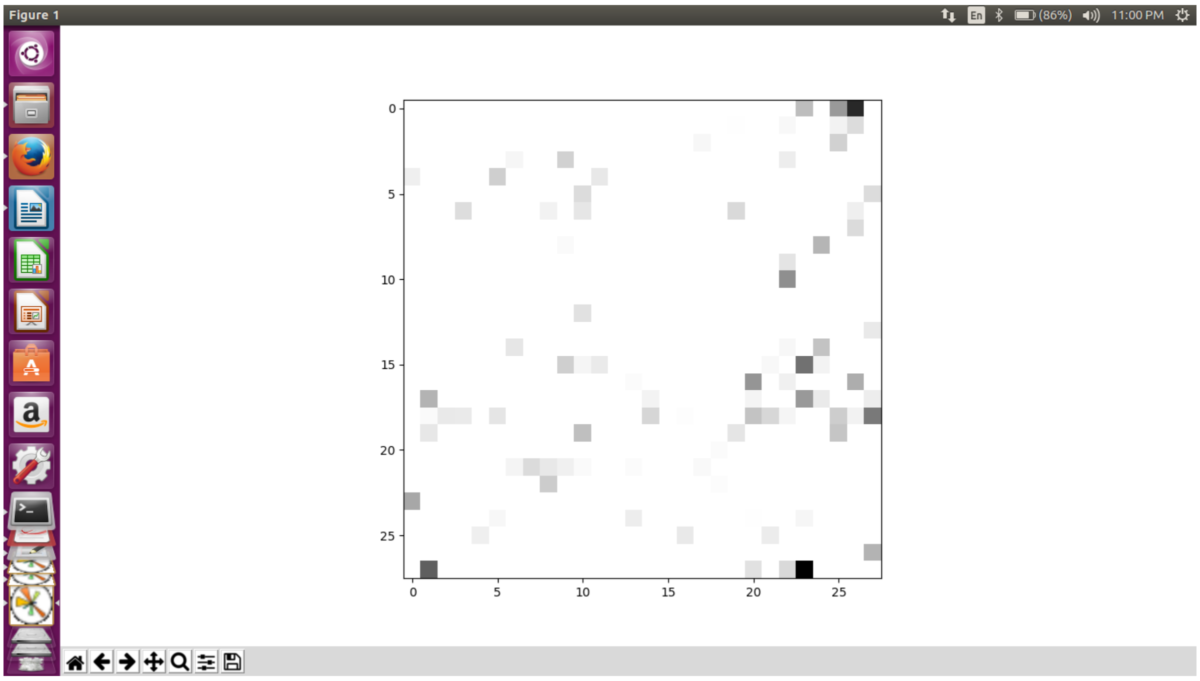
Using the standard model for L1 regularization with C (inverse of lambda) = 1, the accuracy, precision, recall and f1 scores are 0.940391459075, 0.94513274336283182, 0.93684210526315792 and 0.94096916299559474, respectively. The weights are again of the order 10^{-3} and 10^{-4}. Also, there are a lot more zeros in the weights vector. The activation map is given below.



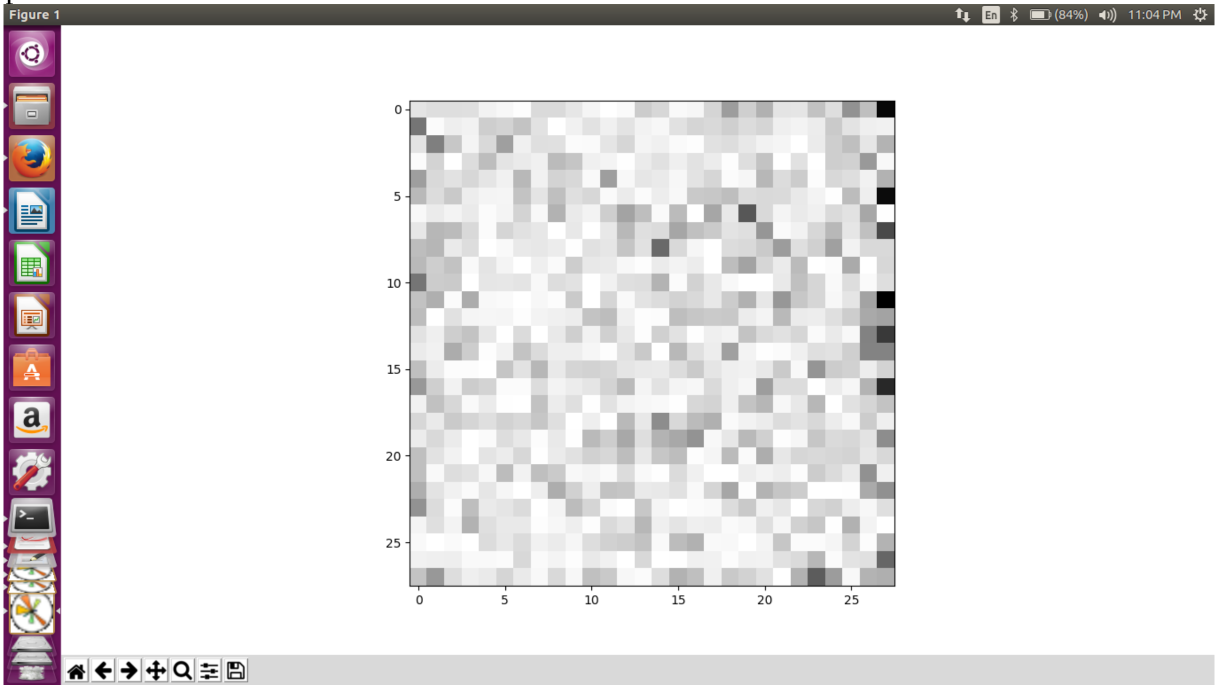
On decreasing C to 0.5, we note that the accuracy, precision, recall and f1 scores become 0.942170818505, 0.94376098418277676, 0.94210526315789478 and 0.94293239683933283, respectively. We can see that the accuracy, recall and f1 scores have increased a bit but the precision has dropped slightly. The weights have decreased in *magnitude*. The activation map is given below.



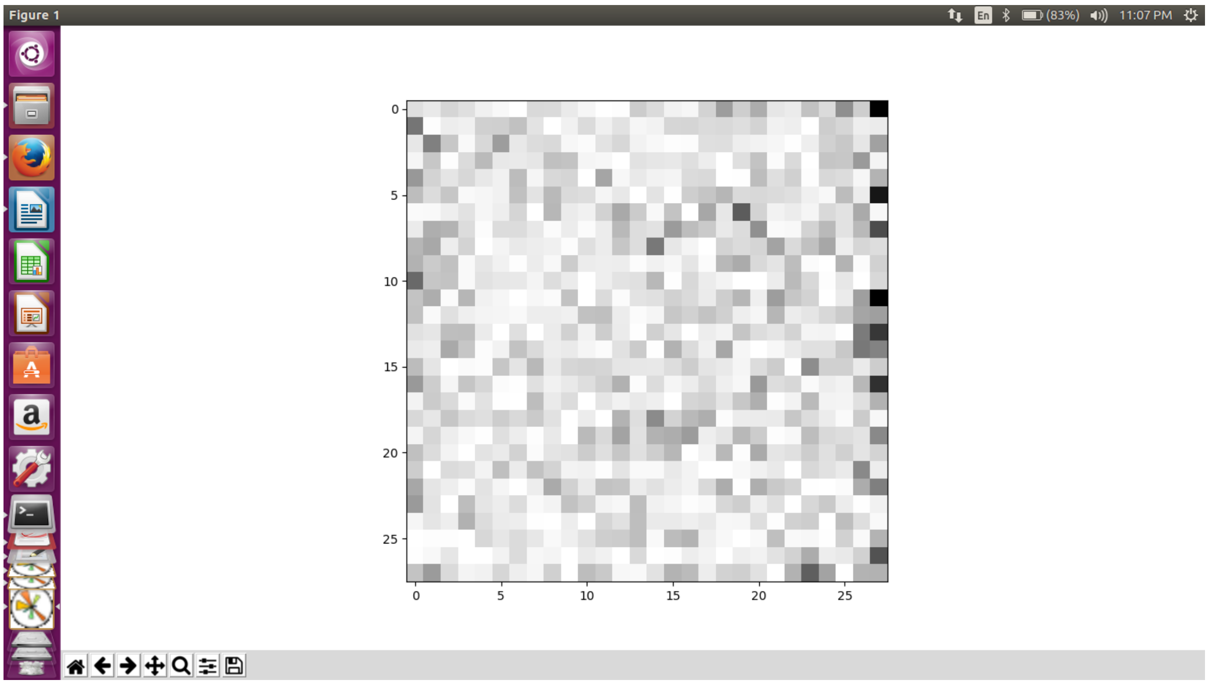
On decreasing C further to 0.001, the accuracy, precision, recall and f1 scores become 0.955516014235, 0.95138888888888884, 0.96140350877192982 and 0.9563699825479931. We can see that all of these have shown an increase. The weights are mostly zero with a few values of powers -2,-3 and -4. The activation map is shown below. We can see how sparse the weights are now.



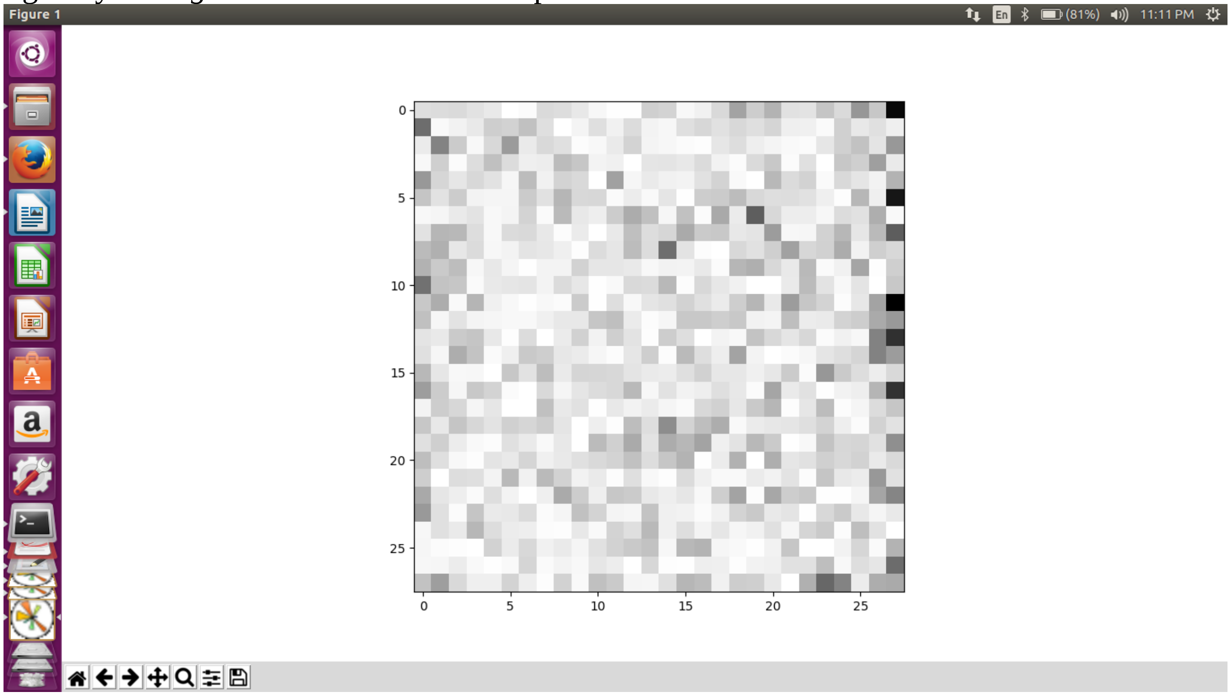
On increasing C to 1.25, the accuracy, precision, recall and f1 scores have become 0.939501779359, 0.94346289752650181, 0.93684210526315792 and 0.94014084507042261. We can see that all these have increased when compared to the case for C = 1. The weights are no more sparse but they have increased in *magnitude* when compared to the case for C = 1. The activation map is shown below.



On increasing C to 1.5, the accuracy, precision, recall and f1 scores have become 0.940391459075, 0.9435626102292769, 0.93859649122807021 and 0.94107299912049258. All of these are lesser than those for the previous case. The weights have increased slightly in *magnitude* as compared to the previous case. The activation map is shown below.



On increasing C to 10, the accuracy, precision, recall and f1 scores become 0.94128113879, 0.94522968197879864, 0.93859649122807021 and 0.94190140845070425. All metrics, except recall which is constant, have increased from the previous case. The weights have increased marginally in *magnitude*. The activation map is shown below.



<u>Observations:</u>

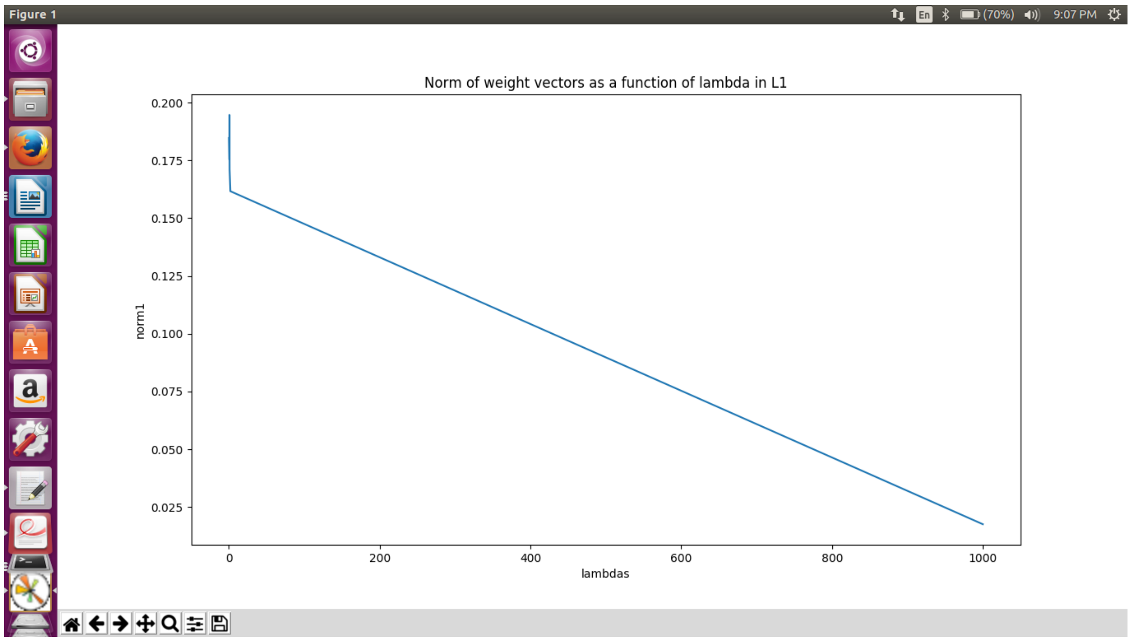
We can see from the activation maps itself that the weights for L2 regularization are larger than those for L1 regularization. We can also note that for small values of C (large values of lambda) in L1 model, the weight vectors contain a lot of zeros and are therefore considered to be more sparse. As we know that L1 model leads to sparsity, the observation holds. Also, for L1 model, with increase in C (lower lambda), the model becomes less and less sparse. We can therefore conclude that increase in lambda leads to less sparse models. Also, in general we can see that for a given C value, the L1 model gives better performance than the L2 model. We can note that L1 models are less stable as compared to L2 models since on running the same code multiple times, the

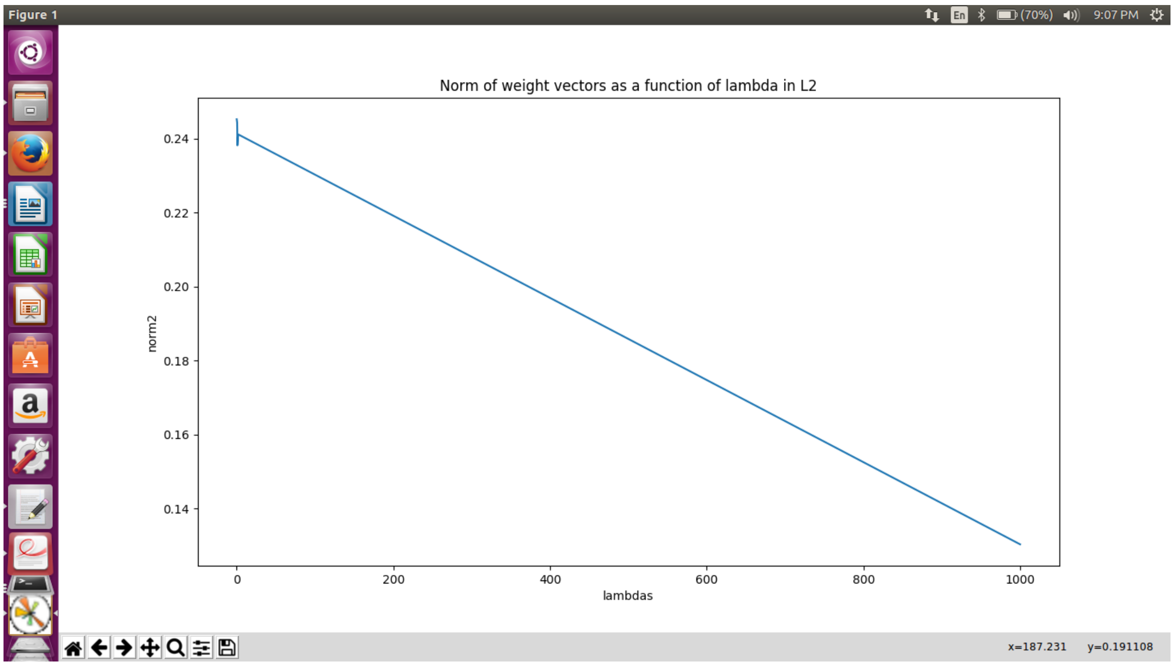
L1 metrics tend to change marginally whereas the L2 metrics remain constant therefore showing that L2 model is stable.

Comparing the norms:

When lambda is 1000, the values of the L1 and L2 norm are 0.017586714163280323 and 0.1302921558748831. When lambda decreases to 2, the norms become 0.16163315668753947 and 0.24108579587684253. The L1 norm has increased by a large value. When lambda = 1, the norms become 0.17109170069477303 and 0.23812004057772188. For lambda = 0.8, the norms become 0.19463959684664556 and 0.24230912462841464. For lambda = 0.666, the norms become 0.17540564969291264 and 0.24429297643063366 whereas for lambda = 0.1, the norms become 0.18468297632180317 and 0.24519525658750993.

We can see that the greatest difference occurs between the norms when lambda is 1000. At this value, C = 0.001 and we have earlier seen that the weights for L1 regularization for such a small value of C are sparse and because of this, the norm for weights of L1 regularization is much smaller than that for L2 regularization. The plots between lambdas and the corresponding norms are given below.





We can see that for the L1 case, there is a sharp change in the beginning thereby showing that for small lambdas, the norm does not change much at first.

Conclusions:

By observing that the L1 model gives more sparse weights when compared to the L2 model, we can conclude that L1 model does produce sparse weights. Also, by judging the performance metrics for L1 and L2 models for a given value of C, we can conclude that L1 model leads to more robust solutions and better performance. Also, since on running the code multiple times, L2 models show constant metrics whereas those for L1 model vary slightly, we can see that L2 model is more stable.